Project Group 13

Members:
Arbman, Kelvin (4943589)
Houterman, Simon (5852471)
Koetsier, Lars (5372739)
Linders, Joris (4947371)

Research Objective

The project explores the relationship between the covid pandemic and road safety in Germany, by indicating how the pandemic influences road traffic, which in turn is expected to have an influence on road safety. The number of vehicles on the road during the pandemic was mainly influenced by governmental measures. The goal of most measures during the pandemic was to reduce contacts and therefore covid cases. But, as this results in less traffic on roads, also road accidents were affected.

In order to determine the relationship between covid data, governmental measures, traffic volume and road safety, several datasets have been used. We concentrated on Germany because accident data is available detailed time-wise. In order to find a relationship between the datasets, a timeline of the covid measures will be made and the relevant data will be plotted on the same graph to get insights into the relations. The conclusions drawn from this project are relevant for the prevention of road traffic accidents in the current and future pandemics.

Research Question:

How was the road safety in Germany affected during the COVID pandemic compared to the previous five years?

Contribution Statement

Be specific. Some of the tasks can be coding (expect everyone to do this), background research, conceptualisation, visualisation, data analysis, data modelling

Everyone: Brainstorming about the assignment and supporting eachother in finding solutions to the problem that is being treated at that moment.

Kelvin Arbman (4943589): Plotting covid data, making the covid timeline.

Simon Houterman (5852471): Collecting road safety data and data for the covid timeline.

Lars Koetsier (5372739): Opening and formatting all the csv files (pipelines), plotting the road safety data, plotting the different data in one graph.

Joris Linders (4947371): Collecting covid data, supporting in plotting covid data and in plotting the different data in one graph.

Data Used

- Road safety-data in Germany
  - Incidents resulting in human injury
    - Destatis Statistiches Bundesamt: Unfälle (polizeilich erfasste): Deutschland, Monate, Unfallkategorie, Ortslage. 2022. URL: https://www-genesis.destatis.de/genesis//online?operation=table&code=46241-0002&bypass=true&levelindex=0&levelid=1667989914858#abreadcrumb (https://www-genesis.destatis.de/genesis//online?operation=table&code=46241-0002&bypass=true&levelindex=0&levelid=1667989914858#abreadcrumb), last visit 2022-11-09
- COVID-data in Germany
  - Positive tests
    - Robert Koch Institut: COVID-19-Fälle nach Altersgruppe und Meldewoche (Tabelle wird jeden Donnerstag aktualisiert). 2022. URL: https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Daten/Altersverteilung.html?nn=13490888 (https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Daten/Altersverteilung.html?nn=13490888), last visit 2022-11-09
  - People on Intensive care (IC)
    - Deutsche Interdisziplinäre Vereinigung für Intensiv- und Notfallmedizin & Robert Koch Institut: Daten zu der Altersstruktur der COVID-ITS-Fälle. 2022. URL: https://www.intensivregister.de/#/aktuelle-lage/downloads (https://www.intensivregister.de/#/aktuelle-lage/downloads), last visit 2022-11-09
  - COVID-related deaths
    - Robert Koch Institut: Coronavirus SARS-CoV-2 Todesfälle nach Sterbedatum. 2022. URL: https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Projekte_RKI/COVID-19_Todesfaelle.html?nn=13490888 (https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Projekte_RKI/COVID-19_Todesfaelle.html?nn=13490888), last visit 2022-11-09
- Traffic data Germany
  - Road usage
    - Bundesanstalt für Strassenwesen: Automatische Zählstellen auf Autobahnen und Bundesstraßen. 2021. URL: https://www.bast.de/DE/Verkehrstechnik/Fachthemen/v2-verkehrszaehlung/Stundenwerte.html;jsessionid=029E1B441B2B51FABA218CEC1B383542.live11311?nn=1819490 (https://www.bast.de/DE/Verkehrstechnik/Fachthemen/v2-verkehrszaehlung/Stundenwerte.html;jsessionid=029E1B441B2B51FABA218CEC1B383542.live11311?nn=1819490), last visit 2022-11-09
- Timespan: five years before pandemic - 2021 (mid-pandemic)

Notebook Overview

1. Imported Libraries

In [1]:
```
import pandas as pd
import numpy as np
import math
import scipy
from scipy.stats.stats import pearsonr

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

import plotly.io as pio

import datetime
import matplotlib.dates as mdates
```

2. Data Pipelines

This section covers the loading of the data into the notebook and data manipulation.

2.1 Road Safety Data Germany

In [2]:
```
# File path
file_path = 'data/road safety germany.csv'

# Open File
df_road_safety_germany = pd.read_csv(file_path,  delimiter=';')

# Adjust Data

# Step 1: This file has the columns Year (YYYY) & Month (mmm)(de), for convience we will add a date column as the tab
#         This column will contain the date of the first day of the corresponding month.
# - Step 1.1: Convert the month values (mmm)(de) (e.i. Januar, Februar, März, ..) to month (mm) values (i.e. 01, 02,
df_road_safety_germany = df_road_safety_germany.replace(['Januar', 'Februar', 'März', 'April', 'Mai', 'Juni', 'Juli',
                                                         'August', 'September', 'Oktober', 'November', 'Dezember'],
                                                        ['01','02','03','04','05','06','07','08','09','10','11','12']

# - Step 1.2: Add [Date] column to dataframe
df_road_safety_germany['Date'] = '01-' + df_road_safety_germany['Month'] + '-' + df_road_safety_germany['Year'].astyp
# - Convert string to date value
df_road_safety_germany['Date'] = pd.to_datetime(df_road_safety_germany['Date'], format = '%d-%m-%Y')
# - Set date column as index
df_road_safety_germany.set_index('Date', inplace=True)

# Step 2: Clean Data, this dataframe contains '...' values for future dates since there is no data available yet.
#         These values will be replaced by empty values
df_road_safety_germany = df_road_safety_germany.replace({'...': None})
```

```
# Step 3: Select only the necesarry columns from the dataframe - We're only interested in the data relating to human
df_road_safety_germany = df_road_safety_germany[['Unfälle mit Personenschaden - Insgesamt',
                                                 'Schwerwiegende Unfälle mit Sachschaden i.e.S - Insgesamt',
                                                 'Sonst. Unfälle unter dem Einfluss berausch. Mittel - Insgesamt',
                                                 'Übrige Sachschadensunfälle - Insgesamt',
                                                 'Insgesamt - Insgesamt']]

# Step 4: Rename column names from German to English
df_road_safety_germany = df_road_safety_germany.rename(columns={
    'Unfälle mit Personenschaden - Insgesamt': 'Accidents involving human injury',
    'Schwerwiegende Unfälle mit Sachschaden i.e.S - Insgesamt': 'Serious accidents with material damange',
    'Sonst. Unfälle unter dem Einfluss berausch. Mittel - Insgesamt': 'Accidents under the influence of toxins',
    'Übrige Sachschadensunfälle - Insgesamt': 'Other accidents',
    'Insgesamt - Insgesamt': 'Total accidents'})

# Step 5: Convert to int
#         First drop future dates without values
df_road_safety_germany = df_road_safety_germany.dropna()
df_road_safety_germany = df_road_safety_germany.astype({'Accidents involving human injury': 'int64'})

# Read File
df_road_safety_germany
```

Out[2]:

Date	Accidents involving human injury	Serious accidents with material damange	Accidents under the influence of toxins	Other accidents	Total accidents
2011-01-01	16448	7045	1142	151332	175967
2011-02-01	16227	6138	1043	137738	161146
2011-03-01	21569	5919	1114	154508	183110
2011-04-01	26411	5717	1245	156631	190004
2011-05-01	30831	6099	1291	170684	208905
...	...	...	...	...	...
2022-01-01	16819	5731	1118	152944	176612
2022-02-01	16067	4889	1110	148594	170660
2022-03-01	21398	4457	1184	164299	191338
2022-04-01	20758	5082	1206	168157	195203
2022-05-01	27159	4397	1139	181329	214024

137 rows × 5 columns

2.2 Incidents Resulting in Human Injury Germany

In [3]:
```
# File path
file_path = 'data/road safety germany.csv'
# Open File
df_accidents_human_injury = pd.read_csv(file_path,  delimiter=';')

# Adjust Data

# Step 1: This file has the columns Year (YYYY) & Month (mmm)(de), for convience we will add a date column as the tab
#         This column will contain the date of the first day of the corresponding month.
# - Step 1.1: Convert the month values (mmm)(de) (e.i. Januar, Februar, März, ..) to month (mm) values (i.e. 01, 02,
df_accidents_human_injury = df_accidents_human_injury.replace(['Januar', 'Februar', 'März', 'April', 'Mai', 'Juni',
                                                               'August', 'September', 'Oktober', 'November', 'Dezember'],
                                                              ['01','02','03','04','05','06','07','08','09','10','11','12']

# - Step 1.2: Add [Date] column to dataframe
df_accidents_human_injury['Date'] = '01-' + df_accidents_human_injury['Month'] + '-' + df_accidents_human_injury['Yea
# - Convert string to date value
df_accidents_human_injury['Date'] = pd.to_datetime(df_accidents_human_injury['Date'], format = '%d-%m-%Y')
# - Set date column as index
df_accidents_human_injury.set_index('Date', inplace=True)

# Step 2: Clean Data, this dataframe contains '...' values for future dates since there is no data available yet.
#         These values will be replaced by empty values
df_accidents_human_injury = df_accidents_human_injury.replace({'...': None})



# Step 3: Select only the necesarry columns from the dataframe - We're only interested in the data relating to human
df_accidents_human_injury = df_accidents_human_injury[['Unfälle mit Personenschaden - innerorts',
                                                        'Unfälle mit Personenschaden - außerorts (ohne Autobahnen)',
                                                        'Unfälle mit Personenschaden - auf Autobahnen',
                                                        'Unfälle mit Personenschaden - Insgesamt',
                                                       ]]

# Step 4: Rename column names from German to English.
df_accidents_human_injury = df_accidents_human_injury.rename(columns={
    'Unfälle mit Personenschaden - Insgesamt': 'Accidents involving human injury - total',
    'Unfälle mit Personenschaden - innerorts': 'Accidents involving human injury - within city limits',
    'Unfälle mit Personenschaden - auf Autobahnen': 'Accidents involving human injury - highway',
    'Unfälle mit Personenschaden - außerorts (ohne Autobahnen)': 'Accidents involving human injury - outside city lim
    })


# Step 5: Convert to int
#         First drop future dates without values
df_accidents_human_injury = df_accidents_human_injury.dropna()
df_accidents_human_injury = df_accidents_human_injury.astype({'Accidents involving human injury - total': 'int64'})
df_accidents_human_injury = df_accidents_human_injury.astype({'Accidents involving human injury - within city limits'
df_accidents_human_injury = df_accidents_human_injury.astype({'Accidents involving human injury - highway': 'int64'})
df_accidents_human_injury = df_accidents_human_injury.astype({'Accidents involving human injury - outside city limits

# Read File
df_accidents_human_injury
```

Out[3]:

Date	Accidents involving human injury - within city limits	Accidents involving human injury - outside city limits, off highway	Accidents involving human injury - highway	Accidents involving human injury - total
2011-01-01	10351	4874	1223	16448
2011-02-01	10791	4296	1140	16227
2011-03-01	15055	5236	1278	21569
2011-04-01	18390	6607	1414	26411
2011-05-01	22273	7017	1541	30831
...	...	...	...	...
2022-01-01	11317	4341	1161	16819
2022-02-01	10884	4011	1172	16067
2022-03-01	15507	4776	1115	21398
2022-04-01	14280	5124	1354	20758
2022-05-01	19568	6206	1385	27159

137 rows × 4 columns

2.3 Positive Covid-19 Tests Germany

In [4]:
```
# Pipeline Covid Positive Test

# File path
file_path = 'data/positive covid tests germany.csv'

# Open File
df_positive_covid_test_germany = pd.read_csv(file_path,  delimiter=';')

# Adjust Data
# Step 1: This file has the column Year_Week (YYYY_ww) for convience we will add a date column as the table index.
#         This column will contain the date of the first day of the corresponding week.
# - Step 1.1: create date column
df_positive_covid_test_germany['Date'] = None
# - Step 1.1: Convert the year and week data to dates using the strptime function.
#             For the year 2020 the week needs to be offset by 7 days, this is due to different interpretations of a
#             week in 2019.
for i in range(len(df_positive_covid_test_germany)):
    d = df_positive_covid_test_germany['Year_Week'][i]
    if d.startswith('2020'):
        df_positive_covid_test_germany['Date'][i] = datetime.datetime.strptime(d + '-1', "%Y_%W-%w") - datetime.timed
    elif d.startswith('2021'):
        df_positive_covid_test_germany['Date'][i] = datetime.datetime.strptime(d + '-1', "%Y_%W-%w")
    else:
        df_positive_covid_test_germany['Date'][i] = datetime.datetime.strptime(d + '-1', "%Y_%W-%w")

# - Set date column as index
df_positive_covid_test_germany.set_index('Date', inplace=True)

# Step 2: Clean Data



# Step 3: Select only the necesarry columns from the dataframe
df_positive_covid_test_germany = df_positive_covid_test_germany[['Gesamt']]

# Step 4: Rename column names from German to English
df_positive_covid_test_germany = df_positive_covid_test_germany.rename(columns={
    'Gesamt': 'Number of Positive COVID-19 tests per 100.000 capita'})

# Step 5: Convert the decimal comma to a decimal point, then convert the string to a float,
df_positive_covid_test_germany["Number of Positive COVID-19 tests per 100.000 capita"]=df_positive_covid_test_germany
df_positive_covid_test_germany['Number of Positive COVID-19 tests per 100.000 capita'] = df_positive_covid_test_germa

# Read File
df_positive_covid_test_germany
```

Out[4]:

Date	Number of Positive COVID-19 tests per 100.000 capita
2020-03-02	1.09
2020-03-09	7.73
2020-03-16	26.91
2020-03-23	40.81
2020-03-30	43.31
...	...
2022-08-29	241.96
2022-09-05	244.93
2022-09-12	274.68
2022-09-19	343.30
2022-09-26	529.81

135 rows × 1 columns

2.4 People on Intensive care (IC) Due to Covid-19 Germany

In [5]:
```
# Pipeline Covid Deaths

# File path
file_path = 'data/covid deaths germany.csv'

# Open File
df_covid_deaths_germany = pd.read_csv(file_path,  delimiter=';')

# Adjust Data
# Step 1: This file has the column Year_Month (YYYY-mm), for convience we will add a date column as the table index.
#         This column will contain the date of the first day of the corresponding month.
df_covid_deaths_germany['Date'] = df_covid_deaths_germany['Year-Month'] + '-01'
# - Convert string to date value
df_covid_deaths_germany['Date'] = pd.to_datetime(df_covid_deaths_germany['Date'], format = '%Y-%m-%d')
# - Set date column as index
df_covid_deaths_germany.set_index('Date', inplace=True)

# Step 2: Clean Data

# Step 3: Select only the necesarry columns from the dataframe
df_covid_deaths_germany = df_covid_deaths_germany[['Number of Covid Deaths']]

# Step 4: Rename column names - not necessary

# Read File
df_covid_deaths_germany
```

Out[5]:

Date	Number of Covid Deaths
2020-03-01	1120
2020-04-01	6069
2020-05-01	1572
2020-06-01	320
2020-07-01	135
2020-08-01	152
2020-09-01	206
2020-10-01	1480
2020-11-01	8604
2020-12-01	22035
2021-01-01	21997
2021-02-01	9803
2021-03-01	5519
2021-04-01	6572
2021-05-01	4734
2021-06-01	1089
2021-07-01	276
2021-08-01	491
2021-09-01	1605
2021-10-01	2491
2021-11-01	8095
2021-12-01	10396
2022-01-01	4520
2022-02-01	5664
2022-03-01	7485
2022-04-01	5232
2022-05-01	2247
2022-06-01	1511
2022-07-01	3294
2022-08-01	2747
2022-09-01	760

2.5 COVID-Related Deaths Germany

In [6]:
```
# Pipeline Covid Intensive Care Cases

# File path
file_path = 'data/intensive care covid cases germany.csv'

# Open File
df_ic_cases_covid_germany = pd.read_csv(file_path,  delimiter=',')


# Adjust Data
# Step 1: Convert date data from datetime to date
for i in range(len(df_ic_cases_covid_germany)):
    d = df_ic_cases_covid_germany['date'][i]
    df_ic_cases_covid_germany['date'][i] = datetime.datetime.strptime(d, '%Y-%m-%dT%H:%M:%S%z').date()


# Step 2: Clean Data

# Step 3: Select only the necesarry columns from the dataframe

# Step 4: Rename column names
df_ic_cases_covid_germany = df_ic_cases_covid_germany.rename(columns={
    'date': 'Date',
    'COVID-19-Fälle': 'Covid Cases on IC'})

df_ic_cases_covid_germany.set_index('Date', inplace=True)
# Read File
df_ic_cases_covid_germany

# ------------ Remark ---------------
# I have no idea why this error is showing, but is the only way I can convert the datetime value to a date value
# ERROR:
    # A value is trying to be set on a copy of a slice from a DataFrame
    #
    # See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
    # df_ic_cases_covid_germany['date'][i] = datetime.strptime(d, '%Y-%m-%dT%H:%M:%S%z').date()
```

```
C:\Users\larsk\AppData\Local\Temp\ipykernel_21928\3601102438.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df_ic_cases_covid_germany['date'][i] = datetime.datetime.strptime(d, '%Y-%m-%dT%H:%M:%S%z').date()
```

Out[6]:

Date	Covid Cases on IC
2020-03-20	200
2020-03-21	308
2020-03-22	364
2020-03-23	451
2020-03-24	616
...	...
2022-10-05	1294
2022-10-06	1344
2022-10-07	1366
2022-10-08	1406
2022-10-09	1453

934 rows × 1 columns

2.6 Traffic Data Germany

The underlying road usage data consists of vehicle detections near detection points on the German road network on an hourly basis. The total size of the different files from 2017 until 2021 adds up to over 14 gigabytes. To ensure that this notebook can operate fairly quickly an addition notebook was used.

This additional notebook named "traffic data pipeline" combines all the data from the different years and different detection points and then groupes that data per day instead of per hour among other data modifications. It then writes that finalised output to a csv file name "traffic data out.csv". That csv file is then opened in this notebook.

In [7]:
```
# File path
file_path = 'traffic data output/traffic data out.csv'

# Open File
df_traffic_data = pd.read_csv(file_path,  delimiter=',')

# Step 4: Rename column names
df_traffic_data = df_traffic_data.rename(columns={
    'Datum': 'Date',
    'Autobahn_KFZ_Total': 'Highway_Vehicle_Count',
    'Autobahn_KFZ_Total_Rolling_Average_7_Days': 'Highway_Vehicle_Count_Moving_Average_7_Days',
    'Bundesstrassen_KFZ_Total': 'Federal_Road_Vehicle_Count',
    'Bundesstrassen_KFZ_Total_Rolling_Average_7_Days': 'Federal_Road_Vehicle_Count_Moving_Average_7_Days',
    'Total': 'Total_Vehicle_Count',
    'Total_Rolling_Average_7_Days': 'Total_Vehicle_Count_Moving_Average_7_Days'
})


# Set index
df_traffic_data.set_index('Date', inplace=True)


# Read File
df_traffic_data
```

Out[7]:

Date	Highway_Vehicle_Count	Highway_Vehicle_Count_Moving_Average_7_Days	Federal_Road_Vehicle_Count	Federal_Road_Vehicle_Count_Moving
2017-01-01	26975627	2.697563e+07	3886847	
2017-01-02	41560509	3.426807e+07	7362853	
2017-01-03	39117647	3.588459e+07	7740331	
2017-01-04	36828052	3.612046e+07	7537968	
2017-01-05	39337185	3.676380e+07	8039175	
...	...	...	...	
2021-12-27	42884833	4.293691e+07	7923371	
2021-12-28	41452599	4.184089e+07	8050999	
2021-12-29	41882438	4.037421e+07	8310961	
2021-12-30	43390035	3.904579e+07	8555059	
2021-12-31	26765605	3.820287e+07	5568421	

1826 rows × 6 columns

3. Data analysis

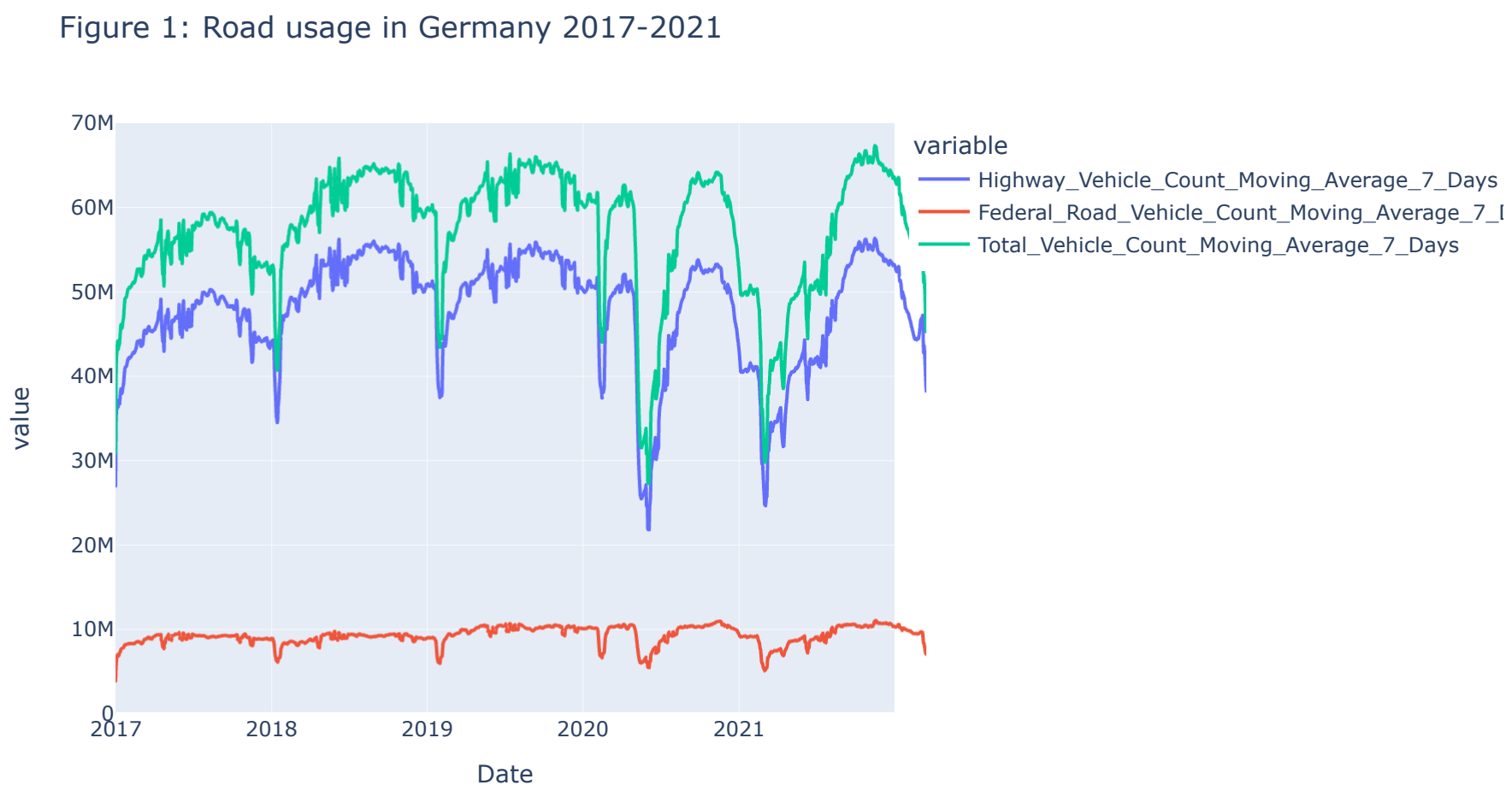
This section covers the data analysis of the selected data files. Each data analysis will be provided with a brief explanation of the most important stand outs.

3.1 Traffic Data Germany

The first part of the data analysis is to look at traffic patterns in Germany. Figure 1 shows the development of road usage in Germany from 2017 until 2021. Here road usage is meassured by the amount of times that a vehicle is detected at a detection point on the German road network. It is opted to show a moving average for 7 days in order to limit the effect of week patterns in this graph (low road usage during the weekends).

Figrue one shows that the road usage on both the highways and federal roads normally peaks during the middle of the year and are normally low during the end and start of a year. It also shows a significant drop in road usage around March 2020 and January 2021. This definitively stands out compared to the previous years and could be a possible result of the COVID-19 meassurements in Germany during this time.

In [8]:
```
fig = px.line(df_traffic_data, x=df_traffic_data.index,
              y= ['Highway_Vehicle_Count_Moving_Average_7_Days',
                  'Federal_Road_Vehicle_Count_Moving_Average_7_Days',
                  'Total_Vehicle_Count_Moving_Average_7_Days'],
              title = "Figure 1: Road usage in Germany 2017-2021")

fig.update_yaxes(range=[0, 70000000])

fig.show()
```

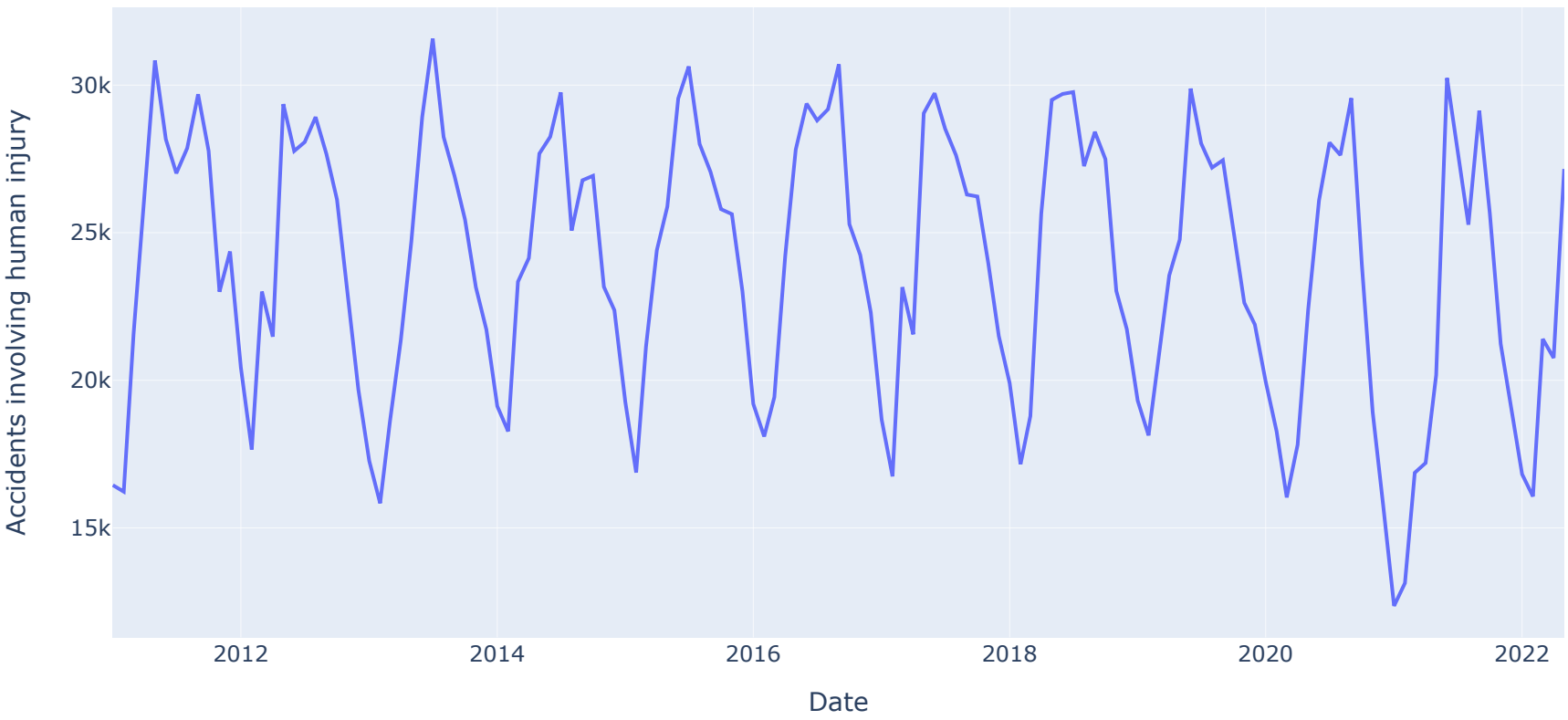


3.2 Road safety Germany

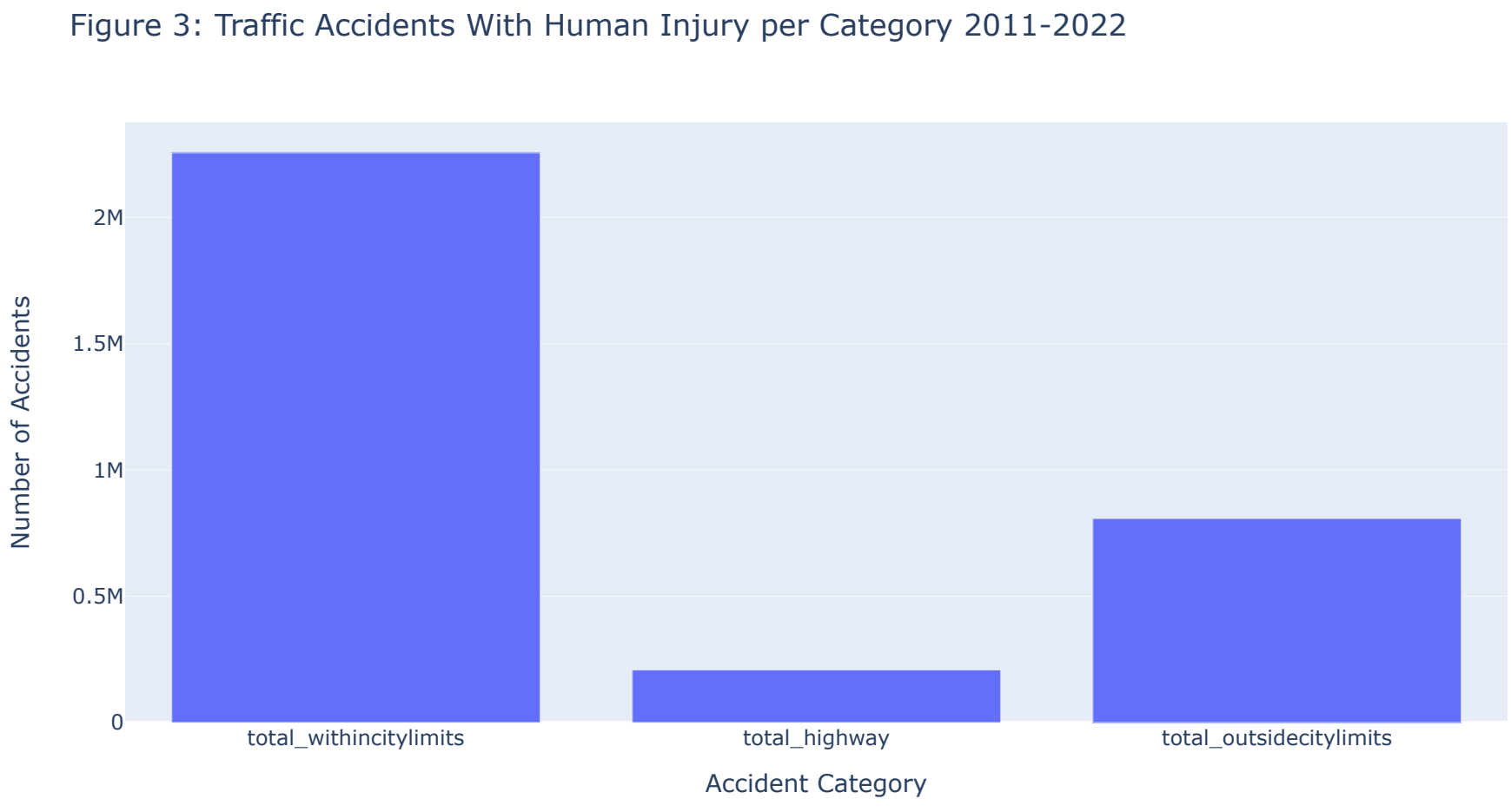
This data analysis shows a graph of the total weekly traffic accidents in Germany over a period of around ten years (Figure 2). This graph clearly shows a seasonal pattern, with more accidents during the summer period and fewer accidents during the winter period.

During the start of 2020 and the start of 2021 there were relatively fewer road accidents during this time of the year compared to the previous years. It also shows that most of the traffic accidents occur within the city limits (Figure 3).

In [9]:
```
# Step 1: select data
df1 = df_road_safety_germany

#Step 2: create line-plot
fig = px.line(df1, x= df_road_safety_germany.index, y= "Accidents involving human injury",
              title = "Figure 2: Weekly total accidents traffic accidents Germany")
fig.show()
```



In [10]:
```
# Step 1: select data
df2 = df_accidents_human_injury

# Step 2: Sum up all the different categories of incidents involving human injuries
total_withincitylimits = 0
for i in df_accidents_human_injury['Accidents involving human injury - within city limits']:
    total_withincitylimits = total_withincitylimits + i

total_highway = 0
for i in df_accidents_human_injury['Accidents involving human injury - highway']:
    total_highway = total_highway + i

total_outsidecitylimits = 0
for i in df_accidents_human_injury['Accidents involving human injury - outside city limits, off highway']:
    total_outsidecitylimits = total_outsidecitylimits + i

#Step 3: Create bar-plot
categories = ['total_withincitylimits', 'total_highway', 'total_outsidecitylimits']
accidents = [total_withincitylimits, total_highway, total_outsidecitylimits]

fig = px.bar(x = categories, y = accidents,
             title = "Figure 3: Traffic Accidents With Human Injury per Category 2011-2022")

fig.update_xaxes(title_text="Accident Category")
fig.update_yaxes(title_text="Number of Accidents")

fig.show()
```



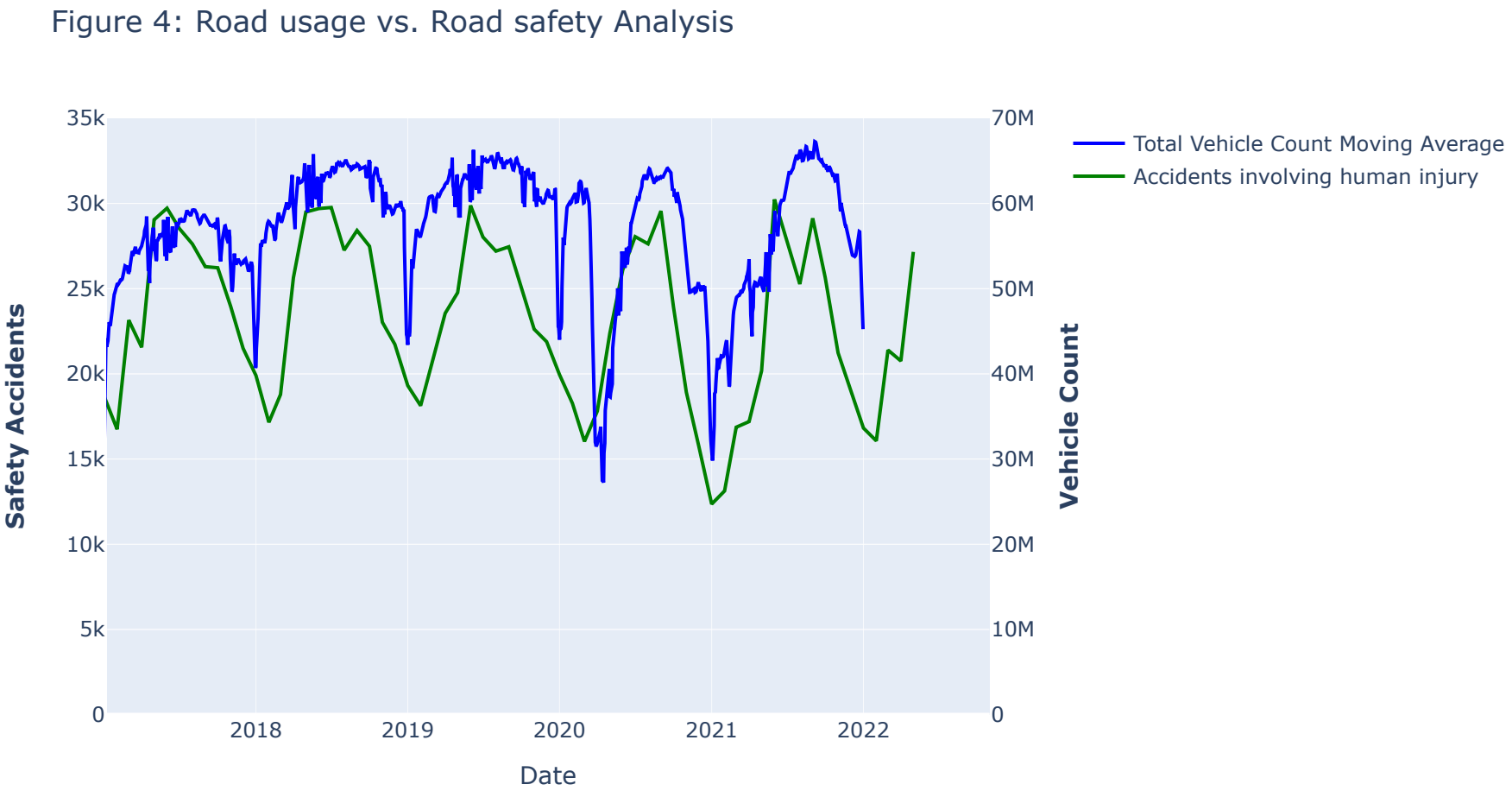
3.3 Road usage vs. road safety Germany

This data analysis shows a graph of the road usage and the total weekly traffic accidents in Germany from 2017 until october 2022 (Figure 4). It is expected that this figure shows a relationship between the road usage and road safety, namely that a decline in road usage results in a decline in traffic accidents and vice versa.

This graph clearly supports this expectation. During the start and end of the year when road usage is lower, there are generally less accidents compared to the rest of the year. More importantly it also shows a significant decline in traffic accidents around March 2020 and January 2021, when also the road usage showed a significant decline.

In [11]:
```
# Create figure with secondary y-axis
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Add traces
fig.add_trace(
    go.Line(x = df_traffic_data.index,
           y = df_traffic_data['Total_Vehicle_Count_Moving_Average_7_Days'],
           name = 'Total Vehicle Count Moving Average 7 Days',
            line_color = 'blue'),
    secondary_y=True,
)

fig.add_trace(
    go.Line(x = df_road_safety_germany.index,
            y = df_road_safety_germany['Accidents involving human injury'],
            name = 'Accidents involving human injury',
            line_color = 'green'),
    secondary_y=False,
)
# Add figure title
fig.update_layout(
    title_text="Figure 4: Road usage vs. Road safety Analysis"
)

# Set x-axis title
fig.update_xaxes(title_text="Date")

# Set y-axes titles
fig.update_yaxes(title_text="<b>Safety Accidents</b>", secondary_y=False)
fig.update_yaxes(range=[0,35000], secondary_y=False)
fig.update_yaxes(title_text="<b>Vehicle Count</b>", secondary_y=True)
fig.update_yaxes(range=[0,70000000], secondary_y=True)
fig.update_xaxes(range=[datetime.date(2017, 1, 7), datetime.date(2022, 10, 31)])


#fig.show()
fig.show()
```

```
C:\Users\larsk\anaconda3\envs\TIL6022\lib\site-packages\plotly\graph_objs\_deprecations.py:378: DeprecationWarning:

plotly.graph_objs.Line is deprecated.
Please replace it with one of the following more specific types
  - plotly.graph_objs.scatter.Line
  - plotly.graph_objs.layout.shape.Line
  - etc.
```



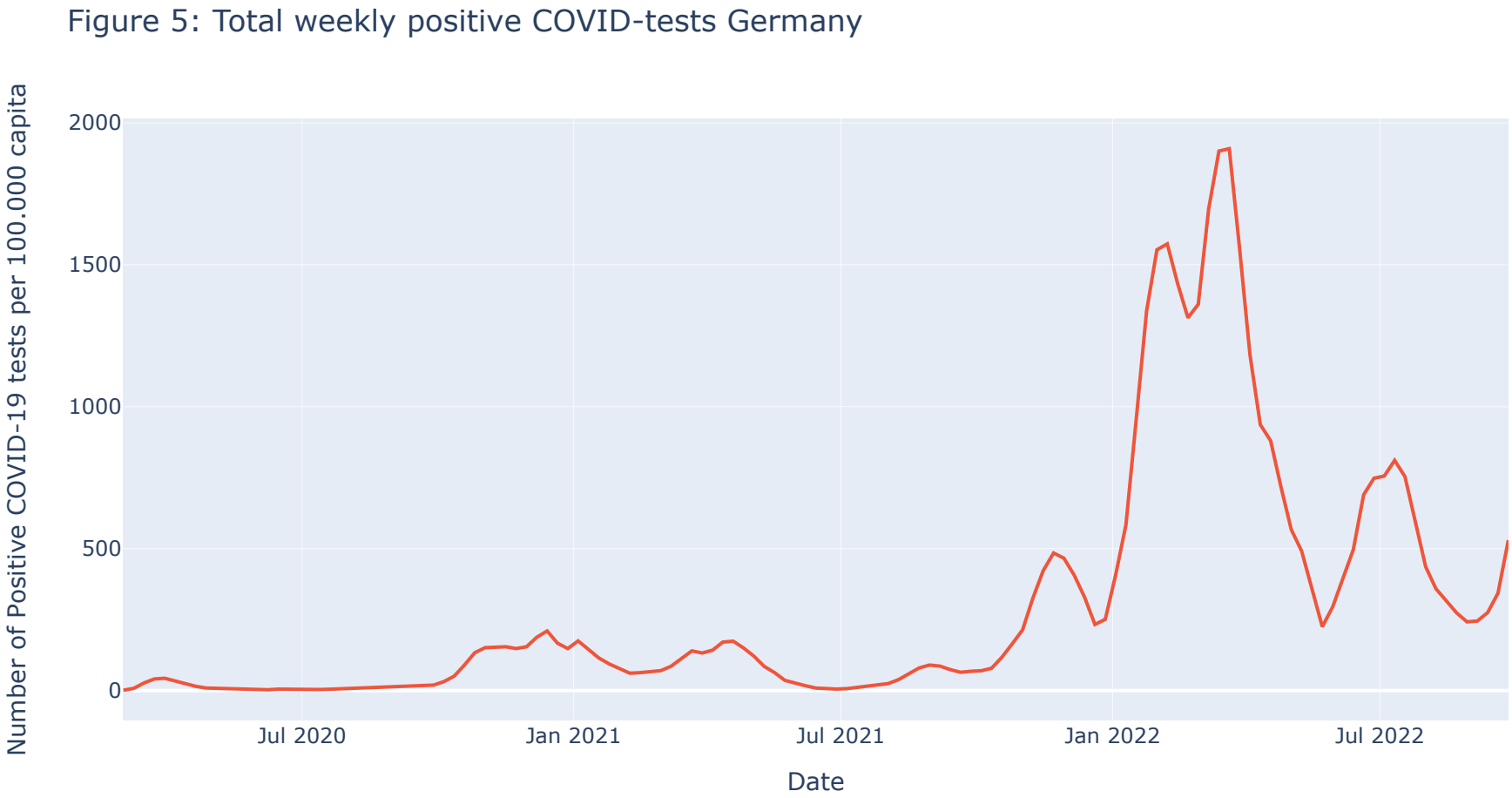
3.4 Covid-19 Germany

This data analysis shows four graphs. The first graph shows the development of the number of weekly positive covid-19 tests per 100.000 capita. This graph also shows a clear spike in positve covid-19 tests during the period from December 2021 till May 2022 (Figure 5). The second graph shows the development of the monthly deaths in Germany related to covid-19. This graph shows a clear spike during the period from November

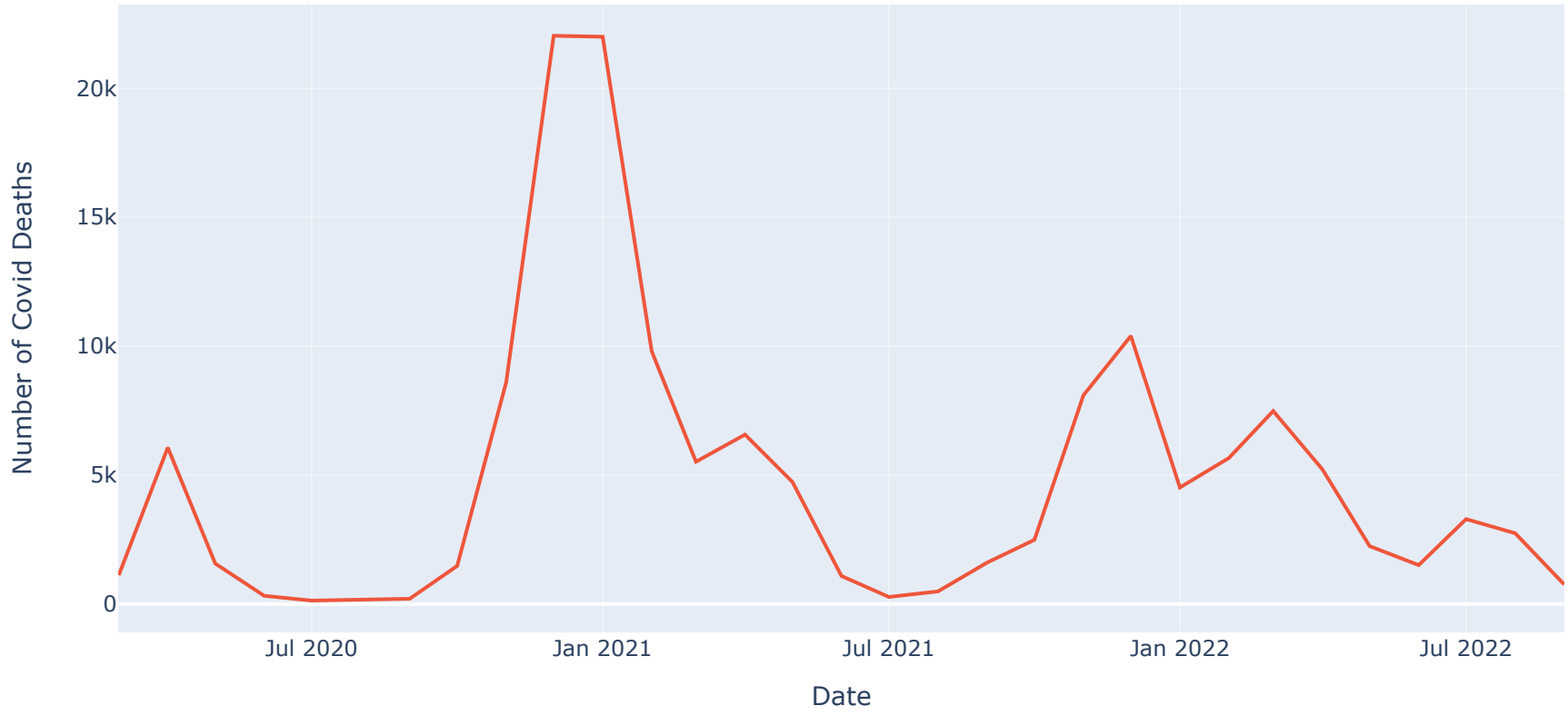
2020 till May 2021 (Figure 6). The third graph shows the development of the daily total covid-19 cases on intensive care. This graph shows a few spikes around April 2020, January 2021, April 2021, and December 2021 (Figure 7).

The fourth graph shows a combination of the three previous graphs. This graph shows that the development of monthly deaths and daily cases on intensive care have a similar behaviour. The development of positive covid-19 tests also has a similar behaviour to these two graphs, but only until January 2022 after which there is a huge spike in positive covid-19 tests. However this spike does not occur with covid-19 deaths or intensive care cases due to covid-19 (Figure 8). This might be due to the fact that testing for covid-19 became more accessible over time. And more testing can result in more positive tests.

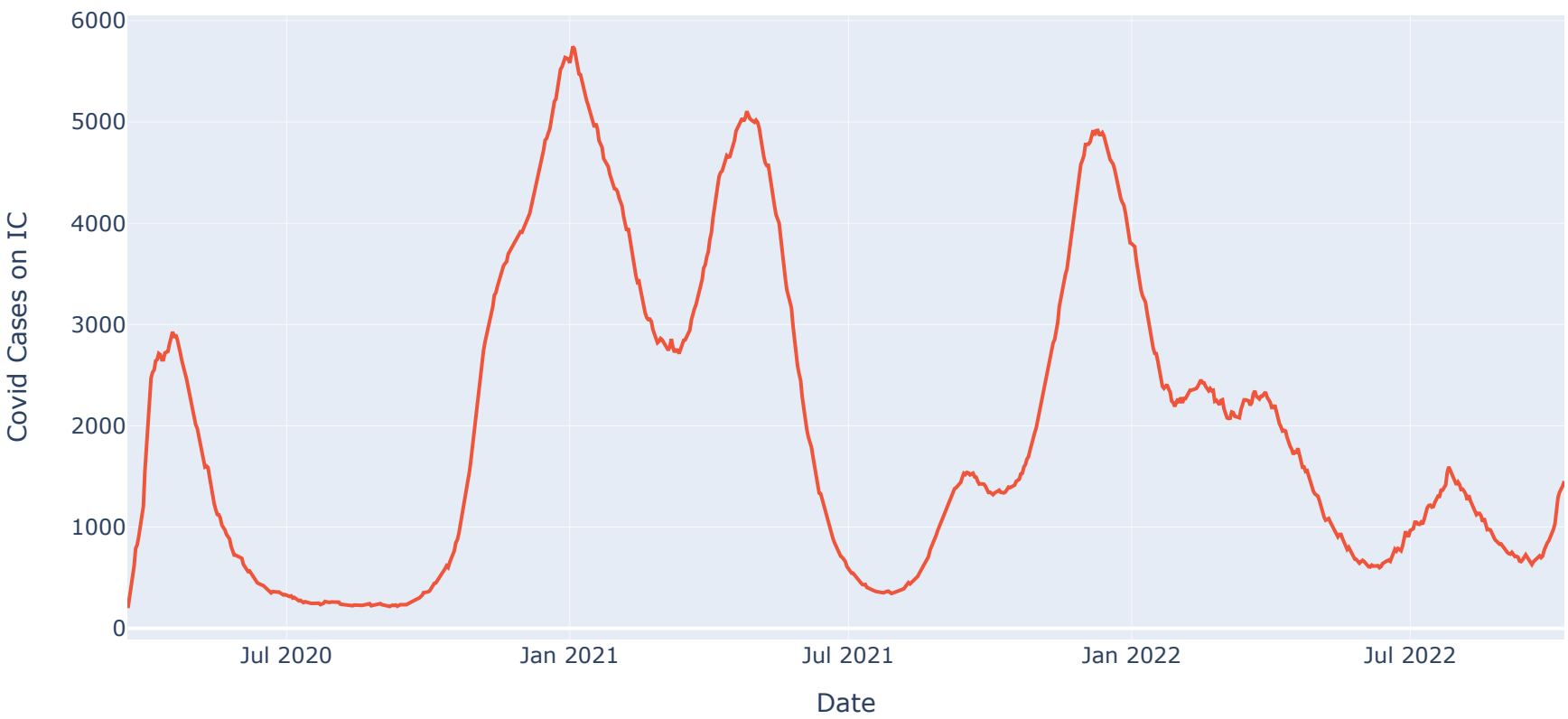
In [12]:
```
# Step 1: Select data
df2 = df_positive_covid_test_germany

# Step 2: Create line-plot
fig = px.line(df2, x= df_positive_covid_test_germany.index,
              y= "Number of Positive COVID-19 tests per 100.000 capita",
              title = "Figure 5: Total weekly positive COVID-tests Germany")
fig.update_traces(line_color='#EF553B')
fig.show()
```



In [13]:
```
# Step 1: Select data
df3 = df_covid_deaths_germany

# Step 2: Create line-plot
fig = px.line(df3, x= df_covid_deaths_germany.index, y= "Number of Covid Deaths",
              title = "Figure 6: Total COVID-Deaths Germany")
fig.update_traces(line_color='#EF553B')
fig.show()
```

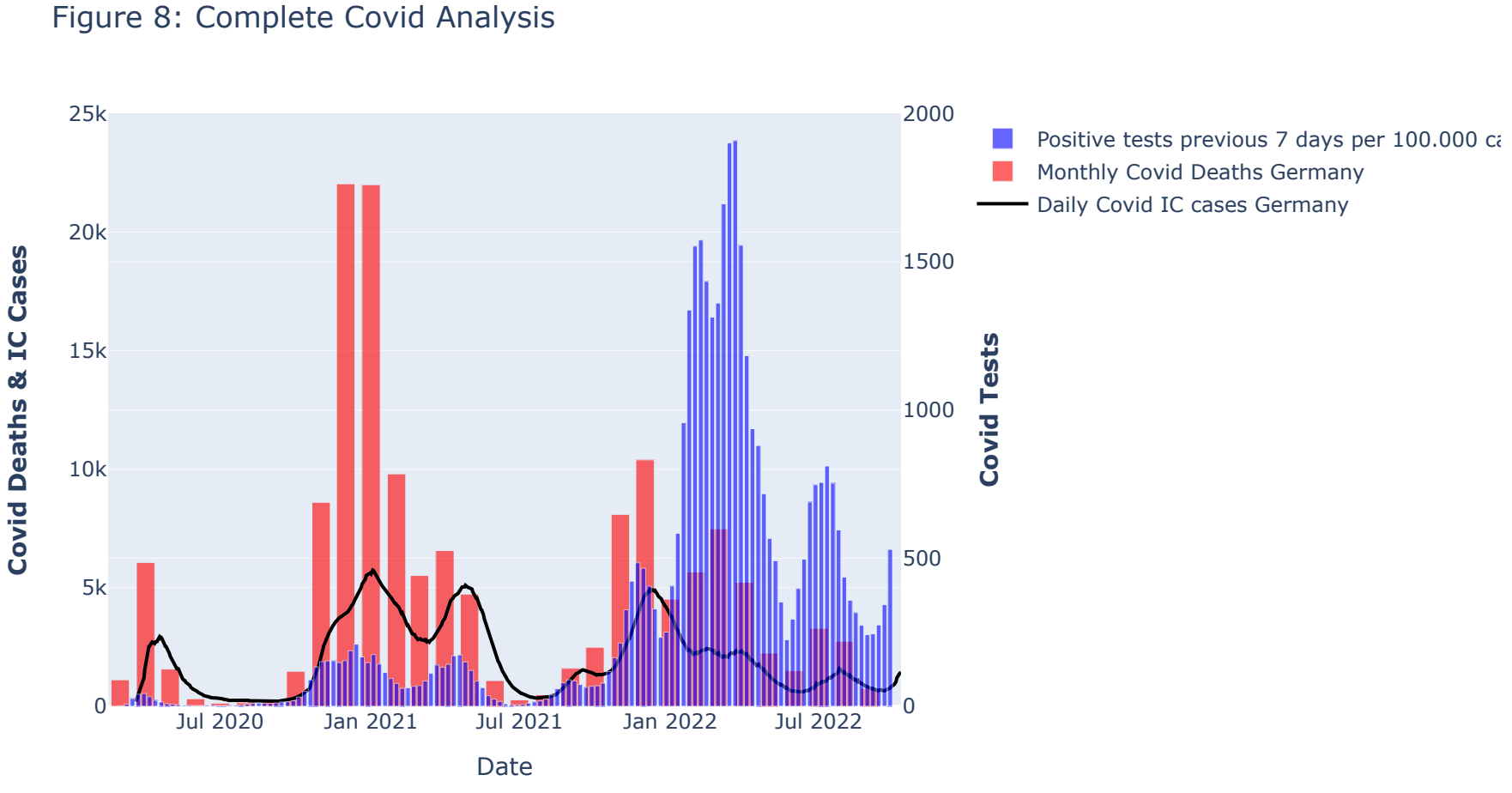


In [14]:
```
# Step 1: Select data
df4 = df_ic_cases_covid_germany

# Step 2: Create line-plot
fig = px.line(df4, x= df_ic_cases_covid_germany.index, y= "Covid Cases on IC",
              title="Figure 7: Total COVID-cases on IC" )
fig.update_traces(line_color='#EF553B')
fig.show()
```



In [15]:
```
# Create figure with secondary y-axis
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Add traces
fig.add_trace(
    go.Bar(x = df2.index,
           y = df2['Number of Positive COVID-19 tests per 100.000 capita'],
           name = 'Positive tests previous 7 days per 100.000 capita',
           marker_color = 'blue',
           opacity = 0.6),
    secondary_y=True,
)

fig.add_trace(
    go.Bar(x = df3.index,
           y = df3['Number of Covid Deaths'],
           name = 'Monthly Covid Deaths Germany',
           marker_color = 'red',
           opacity = 0.6),
    secondary_y=False,
)

fig.add_trace(
    go.Line(x = df4.index,
            y = df4['Covid Cases on IC'],
            name = 'Daily Covid IC cases Germany',
            line_color = 'black'),
    secondary_y=False,
)
# Add figure title
fig.update_layout(
    title_text="Figure 8: Complete Covid Analysis"
)

# Set x-axis title
fig.update_xaxes(title_text="Date")

# Set y-axes titles
fig.update_yaxes(title_text="<b>Covid Deaths & IC Cases</b>", secondary_y=False)
fig.update_yaxes(range=[0,25000], secondary_y=False)
fig.update_yaxes(title_text="<b>Covid Tests</b>", secondary_y=True)
fig.update_yaxes(range=[0,2000], secondary_y=True)

fig.show()
```

```
C:\Users\larsk\anaconda3\envs\TIL6022\lib\site-packages\plotly\graph_objs\_deprecations.py:378: DeprecationWarning:

plotly.graph_objs.Line is deprecated.
Please replace it with one of the following more specific types
  - plotly.graph_objs.scatter.Line
  - plotly.graph_objs.layout.shape.Line
  - etc.
```



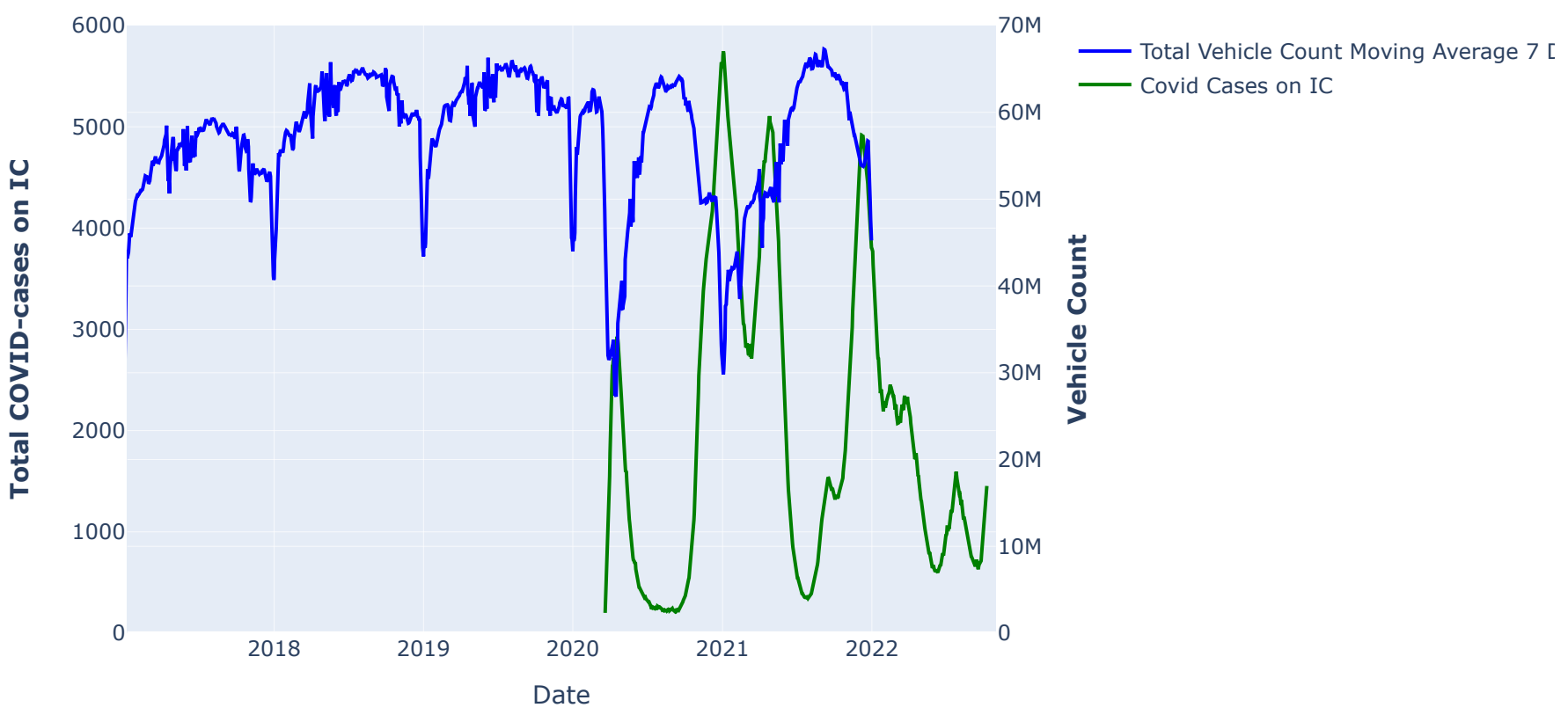
3.5 Road usage vs. Road safety vs. Covid-19 Germany

After establishing that there is a clear relation between the road usage and road safety, it is now time to compare this to the Covid-19 analysis. Now with the expectation that there is relationship between the decline in road usage and road safety and Covid-19. Figure 9 shows the road usage compared to the number of Covid-19 intensive care cases from 2017 until october 2022. The choice to compare with Covid-19 intensive care cases is due to the fact that this was an important indicator for Governments to implement measurements against Covid-19, which in their place might have had an effect on the road usage.

Figure 9 clearly shows that the road usage and Covid-19 analysis are aligned, in the sence that the peaks of the Covid-19 cases are aligned with the low points of the road usage around March 2020 and January 2021. Since the development of the road usage and road safety are also aligned, it can be concluded that there was a relationship between road safety and Covid-19.

In [16]:
```
# Create figure with secondary y-axis
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Add traces
fig.add_trace(
    go.Line(x = df_traffic_data.index,
            y = df_traffic_data['Total_Vehicle_Count_Moving_Average_7_Days'],
            name = 'Total Vehicle Count Moving Average 7 Days',
             line_color = 'blue'),
    secondary_y=True,
)

fig.add_trace(
    go.Line(x = df_ic_cases_covid_germany.index,
            y = df_ic_cases_covid_germany['Covid Cases on IC'],
            name = 'Covid Cases on IC',
            line_color = 'green'),
    secondary_y=False,
)
# Add figure title
fig.update_layout(
    title_text="Figure 9: Road usage vs. Covid-19 Analysis"
)

# Set x-axis title
fig.update_xaxes(title_text="Date")

# Set y-axes titles
fig.update_yaxes(title_text="<b>Total COVID-cases on IC</b>", secondary_y=False)
fig.update_yaxes(range=[0,6000], secondary_y=False)
fig.update_yaxes(title_text="<b>Vehicle Count</b>", secondary_y=True)
fig.update_yaxes(range=[0,70000000], secondary_y=True)
fig.update_xaxes(range=[datetime.date(2017, 1, 7), datetime.date(2022, 10, 31)])


fig.show()
```

Figure 9: Road usage vs. Covid-19 Analysis



3.6 Timeline Animation of Important Stages During the Covid-19 Pandemic

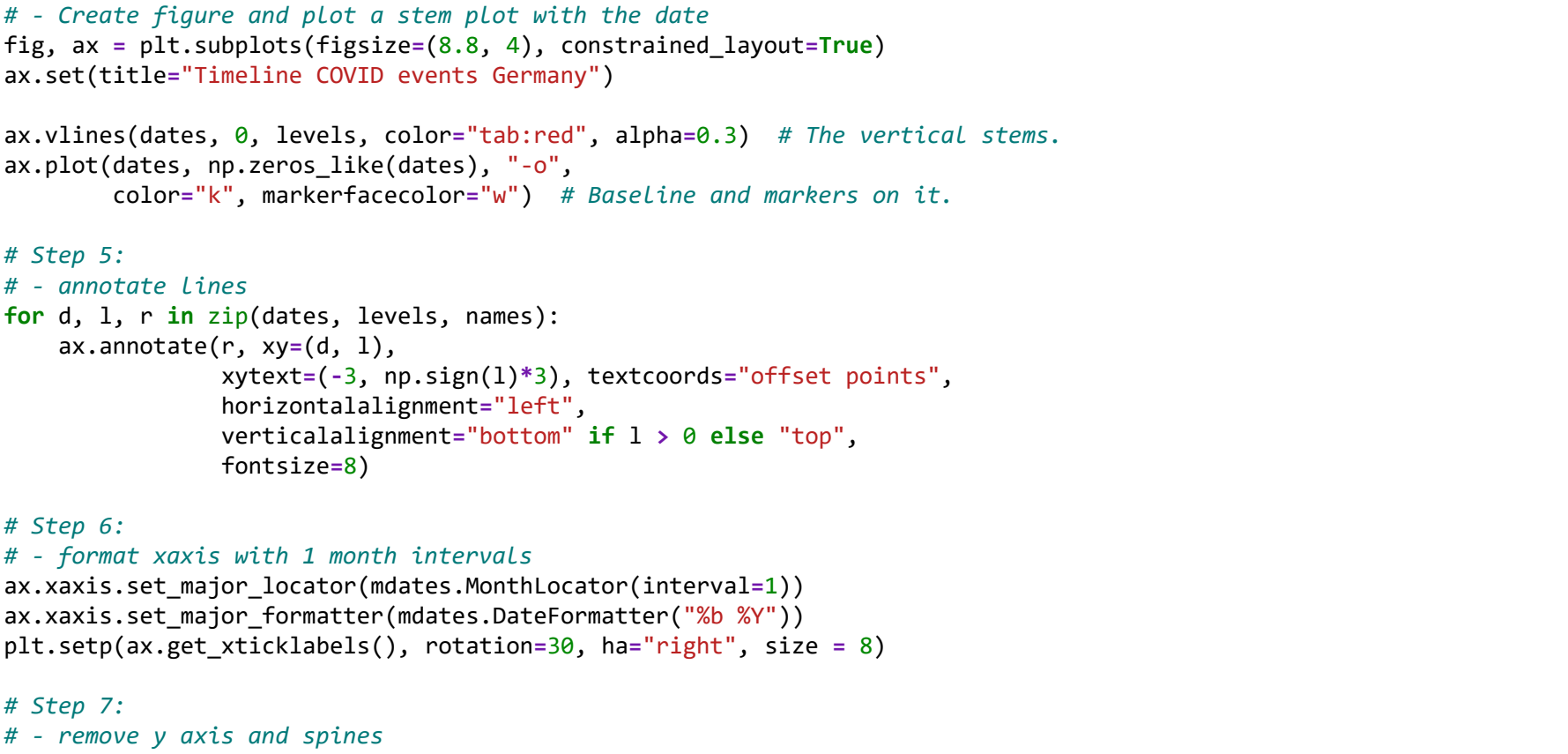
The graph below shows a timeline of the key-events that happened during the COVID-19 pandemic. These events have a relationship with the behaviour of the graphs of the COVID-data that were shown previously. Halfway through april 2020 was the first spike of people on the Intensive

Care. This was around one month after the first big lockdown was announced and implemented by the German government. Right after the peak was over, the government decided that the first ease of measures were ready to be implemented.

At the end of 2020, another big COVID-wave started, which resulted into a lot of deaths and people on the IC. People didn't test a lot, which was also due to the fact that there wasn't as much of testing capacity as there was later on. At the end of 2020, the vaccination campagne started. This did not have an immediate effect on the widespread of COVID, because it took a lot of time before everyone was vaccinated. Eventually, after the majority of the people were vaccinated against COVID-19, the number of people who died from the virus did decrease in the year after. During the winter of 2021/2022, the Omikron variant presented itself more frequently among the people, leading to a lot of people testing positive on COVID. However, the government did not take too many measures, because not a lot of people died relatively because of this variant. Further easing of the measures for non-vaccinated people also took place after the biggest spike during this winter period.

In [17]:

```
#Step 1:
# - Define two lists:
#    - Names of events
#    - Corresponding dates of events
names = ['First Case China [1]','First Case Germany [1]',
         'First big event cancelled [1]','1st lockdown begins [4]',
         'First ease of measures [1]','Test labs overloaded [1]',
         'New measures [4]','Start Lockdown light [4]','Lockdown tightened [4]',
         'Hard lockdown [4]','Start vaccination campagne [4]',
         'First big ease of measures [4]','Hard lockdown [2]',
         'No more measures if vaccinated [4]','Omikron variant [5]',
         'Less 2G/3G measures [3]', 'End of most measures [3]']

dates = ['2019-12-31', '2020-01-27', '2020-03-04',
         '2020-03-22', '2020-04-20', '2020-08-25',
         '2020-10-14', '2020-10-28', '2020-11-25',
         '2020-12-12', '2020-12-27', '2021-03-08',
         '2021-03-23', '2021-05-09', '2021-12-27',
         '2022-02-01', '2022-03-20']

# Step 2:
# - Convert date strings (e.g. 2014-10-18) to datetime
dates = [datetime.datetime.strptime(d, "%Y-%m-%d") for d in dates]

# Step 3:
# - Choose some  levels
levels = np.tile([-5, 5, -3, 3, -1, 1],
                 int(np.ceil(len(dates)/6)))[:len(dates)]

# Step 4:
# - Create figure and plot a stem plot with the date
fig, ax = plt.subplots(figsize=(8.8, 4), constrained_layout=True)
ax.set(title="Timeline COVID events Germany")

ax.vlines(dates, 0, levels, color="tab:red", alpha=0.3)  # The vertical stems.
ax.plot(dates, np.zeros_like(dates), "-o",
        color="k", markerfacecolor="w")  # Baseline and markers on it.

# Step 5:
# - annotate lines
for d, l, r in zip(dates, levels, names):
    ax.annotate(r, xy=(d, l),
                xytext=(-3, np.sign(l)*3), textcoords="offset points",
                horizontalalignment="left",
                verticalalignment="bottom" if l > 0 else "top",
                fontsize=8)

# Step 6:
# - format xaxis with 1 month intervals
ax.xaxis.set_major_locator(mdates.MonthLocator(interval=1))
ax.xaxis.set_major_formatter(mdates.DateFormatter("%b %Y"))
plt.setp(ax.get_xticklabels(), rotation=30, ha="right", size = 8)

# Step 7:
# - remove y axis and spines
# - set margins
ax.yaxis.set_visible(False)
ax.spines[["left", "top", "right"]].set_visible(False)
ax.margins(y=0.2)

plt.show()
```



3.6.1 References for Timeline

[1] Mitteldeutscher Rundfunk (MDR): 2020: Die Chronik der Corona-Krise. 2020. URL: https://www.mdr.de/nachrichten/jahresrueckblick/corona-chronik-chronologie-coronavirus-102.html (https://www.mdr.de/nachrichten/jahresrueckblick/corona-chronik-chronologie-coronavirus-102.html), last visit 2022-11-08

[2] Mitteldeutscher Rundfunk (MDR): Die Chronik der Corona-Krise 2021. 2022. URL: https://www.mdr.de/nachrichten/jahresrueckblick/corona-nachrichten-jahresrueckblick-chronologie-100.html (https://www.mdr.de/nachrichten/jahresrueckblick/corona-nachrichten-jahresrueckblick-chronologie-100.html), last visit 2022-11-08

[3] Mitteldeutscher Rundfunk (MDR): Die Chronik der Corona-Krise 2022. 2022. URL: https://www.mdr.de/nachrichten/jahresrueckblick/corona-chronologie-jahresrueckblick-102.html (https://www.mdr.de/nachrichten/jahresrueckblick/corona-chronologie-jahresrueckblick-102.html), last visit 2022-11-08

[4] Handelsblatt: Corona-Chronik. Bundesregierung bestellt 80 Millionen Dosen Omikron-Impfstoff bei Biontech. 2021. URL: https://www.handelsblatt.com/politik/corona-chronik-bundesregierung-bestellt-80-millionen-dosen-omikron-impfstoff-bei-biontech/25584942.html (https://www.handelsblatt.com/politik/corona-chronik-bundesregierung-bestellt-80-millionen-dosen-omikron-impfstoff-bei-biontech/25584942.html), last visit 2022-11-08

[5] Robert-Koch-Institut (RKI): Anzahl und Anteile von VOC und VOI in Deutschland (xlsx). 2022. URL: https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Daten/VOC_VOI_Tabelle.xlsx?__blob=publicationFile (https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Daten/VOC_VOI_Tabelle.xlsx?__blob=publicationFile) via https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Virusvariante.html (https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Virusvariante.html), last visit 2022-10-24