The Reactions of the German Stock Market to COVID-19 and Containment Policies: A Vector Autoregressive Analysis

1. Preparing the datasets

1.1 Importing data

In [1]:
```
# Importing the necessary python packages
import investpy   #investpy is a Python package to retrieve data from investing.com
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#retrieving data on the German HDAX performance index that includes all companies from the DAX30, MDAX and TecDAX from
hdax = investpy.get_index_historical_data(index='hdax',
                                          country='germany',
                                          from_date='01/01/2020',
                                          to_date='23/07/2021')
```

In [3]:
```
#saving the HDAX data to a seperate .csv file
hdax.to_csv("hdax-data.csv")
```

In [4]:
```
#loading the dataset from Our World in Data on global Covid-19 statistics
covid_raw = pd.read_csv("owid-covid-data.csv", parse_dates=["date"])
```

In [5]:
```
#loading the dataset from Our World in Data on covid containment and health measures (including the stringency index) b
measures = pd.read_csv("stringency_data/covid-containment-and-health-index.csv", parse_dates=["Day"])
```

1.2 Filtering data

In [6]:

```
#filter and prepare the dataset on Covid-19 cases and deaths in Germany

#drop all unneccesary columns
covid_raw.drop(columns=['iso_code','continent', 'total_deaths', 'new_deaths', 'new_cases_smoothed', 'new_deaths_smoothe
                        'new_cases_per_million', 'new_cases_smoothed_per_million',
                        'total_deaths_per_million', 'new_deaths_per_million',
                        'new_deaths_smoothed_per_million',
                        'reproduction_rate',
                        'icu_patients',
                        'icu_patients_per_million', 'hosp_patients',
                        'hosp_patients_per_million', 'weekly_icu_admissions',
                        'weekly_icu_admissions_per_million', 'weekly_hosp_admissions',
                        'weekly_hosp_admissions_per_million', 'new_tests', 'total_tests',
                        'total_tests_per_thousand', 'new_tests_per_thousand',
                        'new_tests_smoothed', 'new_tests_smoothed_per_thousand',
                        'positive_rate', 'tests_per_case', 'tests_units', 'total_vaccinations',
                        'people_vaccinated', 'people_fully_vaccinated', 'new_vaccinations',
                        'new_vaccinations_smoothed', 'total_vaccinations_per_hundred',
                        'people_vaccinated_per_hundred', 'people_fully_vaccinated_per_hundred',
                        'new_vaccinations_smoothed_per_million','population',
                        'population_density', 'median_age', 'aged_65_older',
                        'aged_70_older', 'gdp_per_capita', 'extreme_poverty',
                        'cardiovasc_death_rate', 'diabetes_prevalence', 'female_smokers',
                        'male_smokers', 'handwashing_facilities', 'hospital_beds_per_thousand',
                        'life_expectancy', 'human_development_index', 'excess_mortality', ], inplace=True)

#filter for data regarding Germany
covid_de = covid_raw[covid_raw["location"]=="Germany"].copy(deep=True)
covid_de.drop(columns=["location"], inplace=True)

#sorting the data by date
covid_de.sort_values(by="date", ascending=True, inplace=True)

#resetting the index to the date
covid_de.set_index("date", inplace=True, drop=True)

#rename the index
covid_de.index.name = "date"
```

In [7]:

```
#filter and prepare the dataset of the containment and health index

#filter for data regarding Germany
measures_de = measures[measures["Entity"]=="Germany"].drop(columns=["Code"]).copy(deep=True)
```

```
measures_de.drop(columns=["Entity"], inplace=True)

#sorting the data by date
measures_de.sort_values(by="Day", inplace=True)

#resetting the index to the date
measures_de.set_index("Day", inplace=True, drop=True)

#rename the index
measures_de.index.name = "date"
```

In [8]:

```
#filter and prepare the dataset on stock index prices

#drop all unneccesary columns
hdax.drop(columns=["High", "Low", "Volume", "Currency"], inplace=True)

#rename columns
hdax.rename(columns={"Open":"hdax_open", "Close":"hdax_close"}, inplace=True)

#rename the index
hdax.index.name = "date"
```

2. Merging the datasets

In [9]:

```
#Do an inner left join of covid data and masures data on hdax data.
#Only dates where price data is available should be included in sample

hdax_covid = hdax.join(covid_de, on="date").copy(deep=True)
hdax_covid_measures = hdax_covid.join(measures_de, on="date").copy(deep=True)
```

3. Dropping missing data

In [10]:

```
#loading the combined data into a new variable and handle missing values
data = hdax_covid_measures.copy(deep=True)
```

In [11]:

```
#data[data.isna().any(axis=1)]
#It seems like data is missing only before the pandemic started, and on the most recent dates.
```

```
#We can easily discard these observations and adjust our sample beginning and end dates
data.dropna(axis=0, how="any", inplace=True)
```

4. Adding Dummy Columns for weekdays

In [12]:

```
#create a column describing what day of week this date is
data['day_of_week'] = data.index.day_name()

#from the new column "day_of_week", create dummy columns filled with 0 and 1
data = data.join(pd.get_dummies(data["day_of_week"])).copy(deep=True)

#rearange the order of the columns
data = data[['hdax_open', 'hdax_close', 'total_cases', 'new_cases',
             'stringency_index', 'containment_index',
             'day_of_week','Monday','Tuesday', 'Wednesday','Thursday','Friday']].copy(deep=True)

#delete the "day_of_week" column
data.drop(columns=["day_of_week"],inplace=True)
```

5. Saving the data for further statistical analysis

In [13]:

```
data.to_csv("prepared_data.csv")
```