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

6. Descriptive Statistics full sample

In [1]:
```
# Importing the necessary python packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import datetime as dt
%matplotlib inline
plt.style.use(['science','no-latex'])
```

In [2]:
```
# Load in the previosuly prepared dataset
df = pd.read_csv("prepared_data.csv", parse_dates=["date"])

# Drop unnecessary columns and dummy columns for this sheet
df = df.loc[:,:"stringency_index"].drop(columns=["hdax_open","total_cases"])

#saving sample dates in variables
sample_1_start = pd.to_datetime("2020-03-03")
sample_1_end = pd.to_datetime("2020-05-30")
sample_2_start = pd.to_datetime("2020-10-01")
sample_2_end = pd.to_datetime("2021-06-30")
```

In [3]:
```
# Generate descriptive statistics
pd.options.display.float_format = "{:.2f}".format
df = df[(df["date"]<=sample_2_end) & (df["date"]>=sample_1_start)]
df.describe()
# df.loc[:,"hdax_close":"stringency_index"].drop(columns=["total_cases"]).describe()
# df.loc[:,"hdax_close":"stringency_index"].drop(columns=["total_cases"]).describe()
```

Out[3]:

	hdax_close	new_cases	stringency_index
count	336.00	336.00	336.00

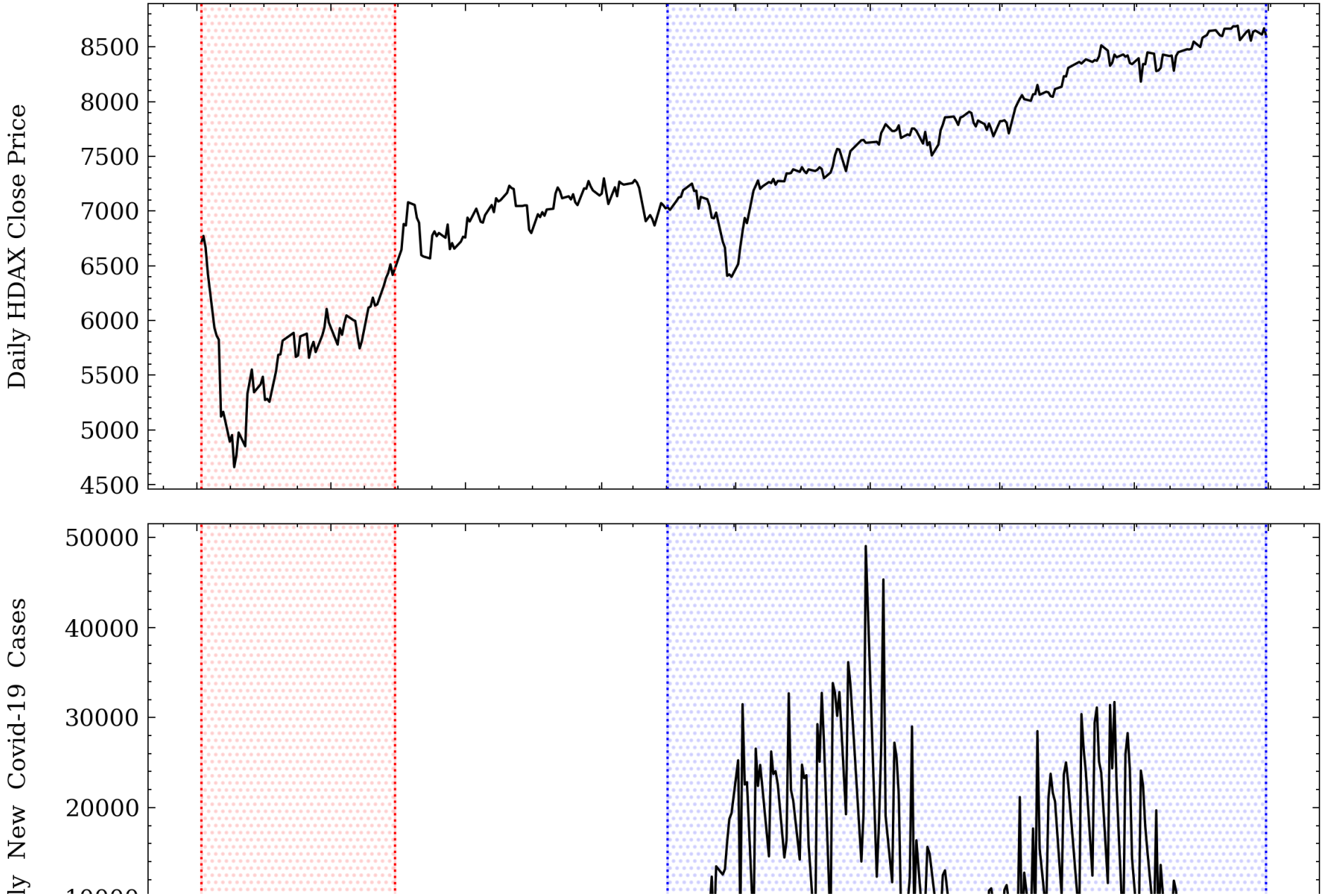
	hdax_close	new_cases	stringency_index
mean	7245.44	8662.63	66.59
std	912.62	9931.07	12.33
min	4658.54	16.00	25.00
25%	6814.13	1154.75	59.72
50%	7231.09	4500.50	67.59
75%	7858.30	13239.75	76.85
max	8694.16	49044.00	85.19

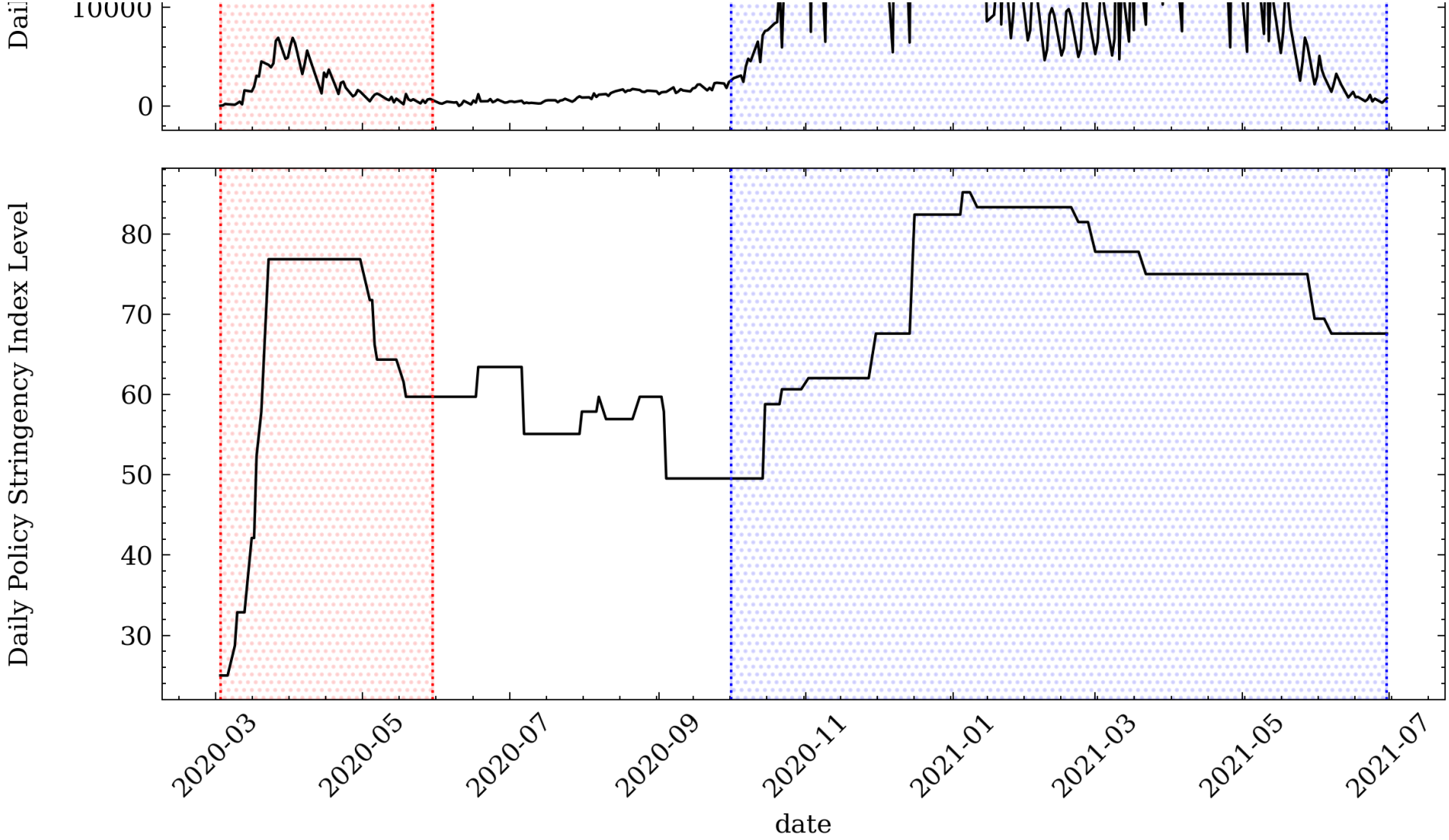
7. Data Plotting and Sample Selection

In [4]:
```
# Loading the column names in a list
columns = list(df.drop(columns=["date"]).columns)
```

In [5]:
```
#plotting the different datapoints and highlighting the two sample areas

fig, axes = plt.subplots(3, 1, tight_layout = True, figsize=(8,10), sharex=True, dpi=300)

#plotting and highlighting samples
for column, ax in zip(columns, axes.reshape(-1)):
    sns.lineplot(x=df.date, y=column, data=df, ax=ax, color="black")
    ax.yaxis.set_label_coords(-0.1,0.5)
    for tick in ax.get_xticklabels():
        tick.set_rotation(45)

    # highlight the first wave sample
    ax.axvline(x = sample_1_start, color = 'red', ls=":")
    ax.axvline(x = sample_1_end, color = 'red', ls=":")
    ax.axvspan(sample_1_start, sample_1_end, alpha=0.15, color='red', hatch="....", fill=False)

    # highlight the second wave sample
    ax.axvline(x = sample_2_start, color = 'blue', ls=":")
    ax.axvline(x = sample_2_end, color = 'blue', ls=":")
    ax.axvspan(sample_2_start, sample_2_end, alpha=0.15, color='blue', hatch="....", fill=False)
```

```
axes[0].set_ylabel("Daily HDAX Close Price")
axes[1].set_ylabel("Daily  New  Covid-19  Cases")
axes[2].set_ylabel("Daily Policy Stringency Index Level");
fig.savefig('all_charts_marked.pdf', format='pdf')
```

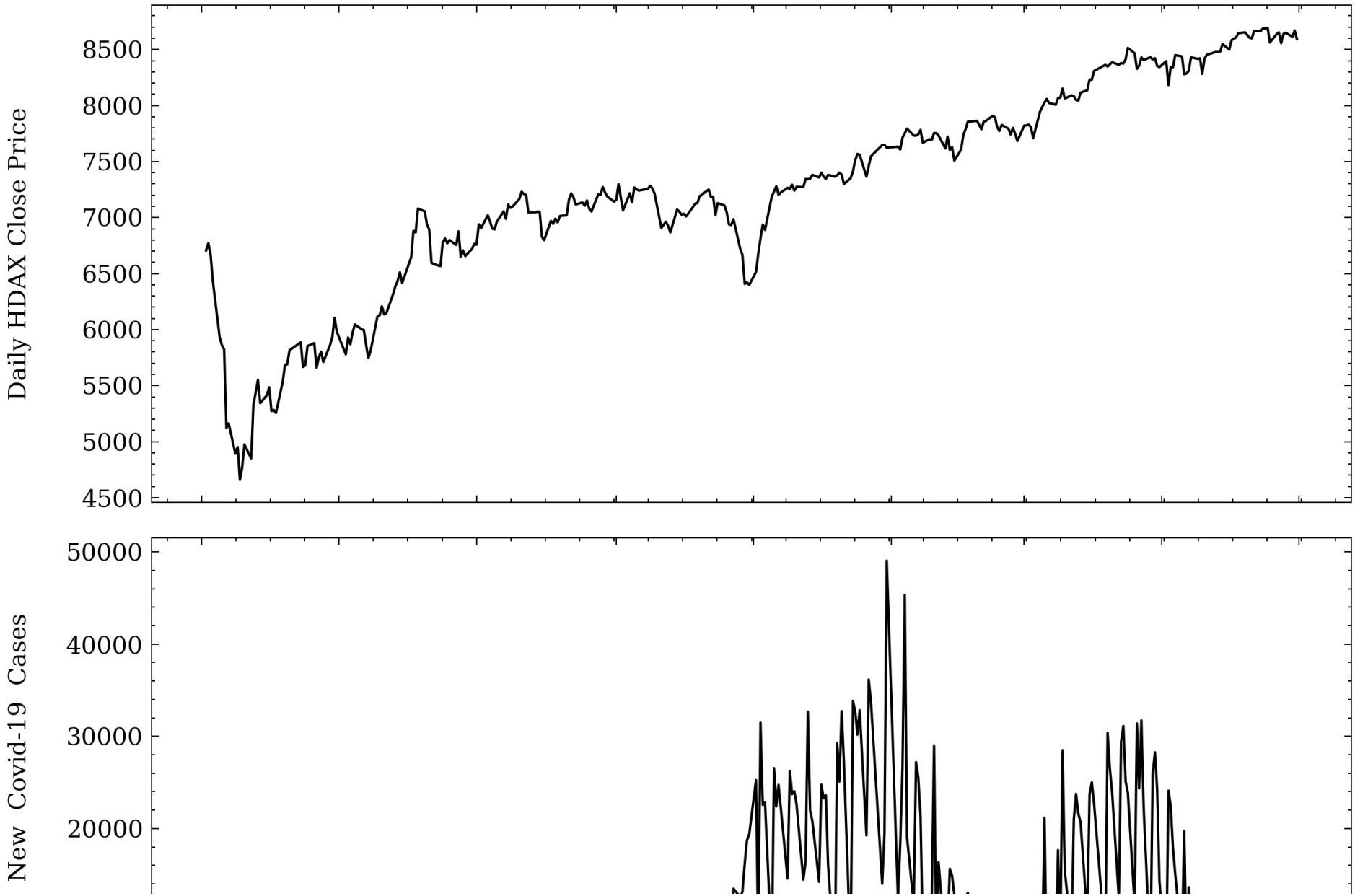


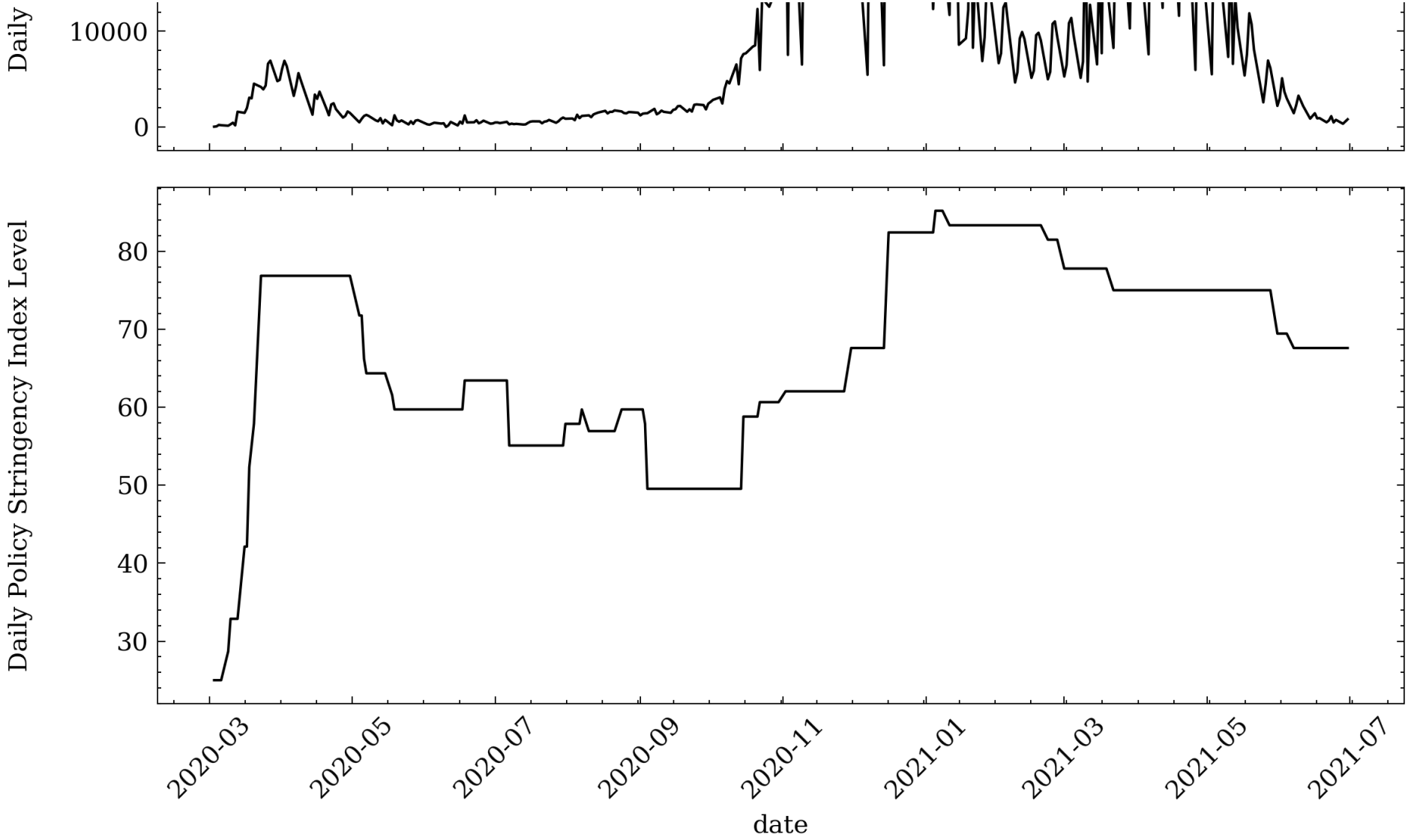


In [6]:

```
#plotting the relevant datapoints without higlighting the sub-samples
fig, axes = plt.subplots(3, 1, tight_layout = True, figsize=(8,10), sharex=True, dpi=300)

#plotting
for column, ax in zip(columns, axes.reshape(-1)):
    sns.lineplot(x=df.date, y=column, data=df, ax=ax, color="black")
    ax.yaxis.set_label_coords(-0.1,0.5)
    for tick in ax.get_xticklabels():
        tick.set_rotation(45)
```

```
axes[0].set_ylabel("Daily HDAX Close Price")
axes[1].set_ylabel("Daily  New  Covid-19  Cases")
axes[2].set_ylabel("Daily Policy Stringency Index Level")
fig.savefig('all_charts.pdf', format='pdf')
```





8. Descriptive Statistics with two sub-samples

In [7]:
```
#generate descriptive statistics of first wave sub-sample
df = pd.read_csv("prepared_data.csv", parse_dates=["date"])
df_1 = df[(df["date"]<=sample_1_end) & (df["date"]>=sample_1_start)].copy(deep=True)
df_1.loc[:,"hdax_close":"stringency_index"].describe()
```

Out[7]:

	hdax_close	total_cases	new_cases	stringency_index
count	61.00	61.00	61.00	61.00
mean	5774.65	107517.39	2146.08	63.01
std	483.03	69802.31	2010.07	16.89
min	4658.54	196.00	37.00	25.00
25%	5484.82	32986.00	600.00	59.72
50%	5823.93	137698.00	1268.00	64.35
75%	6045.76	170588.00	3394.00	76.85
max	6772.21	182922.00	6933.00	76.85

In [8]:

```
#generate descriptive statistics of second wave sub-sample
df_2 = df[(df["date"]<=sample_2_end) & (df["date"]>=sample_2_start)].copy(deep=True)
df_2.loc[:,"hdax_close":"stringency_index"].describe()
```

Out[8]:

	hdax_close	total_cases	new_cases	stringency_index
count	188.00	188.00	188.00	188.00
mean	7831.60	2219262.78	14312.72	72.35
std	569.41	1137482.19	10104.21	9.36
min	6398.20	295539.00	341.00	49.54
25%	7366.11	1189383.00	6117.00	67.59
50%	7803.20	2349821.00	12057.00	75.00
75%	8363.98	3285816.50	22614.50	81.48
max	8694.16	3736205.00	49044.00	85.19

The descriptive statistics clearly show that the first and second waves are very different!

9. Saving both samples seperately for further statistical analysis

In [9]:

```
# Saving first wave sample
df_1.set_index("date", inplace=True, drop=True)
df_1.to_csv("prepared_data_sample_1.csv")

# Saving second wave sample
df_2.set_index("date", inplace=True, drop=True)
df_2.to_csv("prepared_data_sample_2.csv")

# Saving full sample timeframe
df_3 = df[(df["date"]<=sample_2_end) & (df["date"]>=sample_1_start)].copy(deep=True)
df_3.to_csv("prepared_data_sample_full.csv")
```

In [ ]:

In [ ]: