# Black Swan Events and Sentiment Analysis


**Group 1**

David Lewis

Jenny Vu

Anna Zhao

Darcy Davis

# MOTIVATION & SUMMARY

- Black swan events are characterised by their extreme rarity and have potentially severe consequences.
- Past 5 year is a interesting period with 2 extremely rare black swans.
- Black swan events mean great risks and opportunities.



Opinion
Columnist

## Universa's 3,612% Return Is Legit (But With an Asterisk)

Whether you believe the gaudy gains are real or not, it's clear that some "crisis hunters" have a secret sauce that offers value to investors beyond conventional tail-risk strategies.

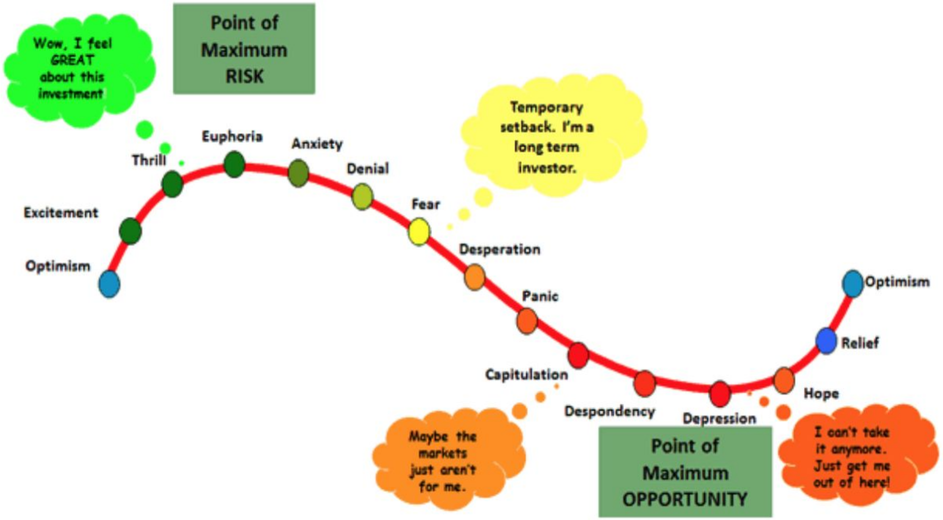
It's raining money at some hedge funds. *Photographer: Mary Turner/Getty Images*

# MOTIVATION & SUMMARY

- If we can predict the black swan events and quantify the panics caused by the events, we can find the point of maximum opportunity.

- Our study can help predict when is the next shock.

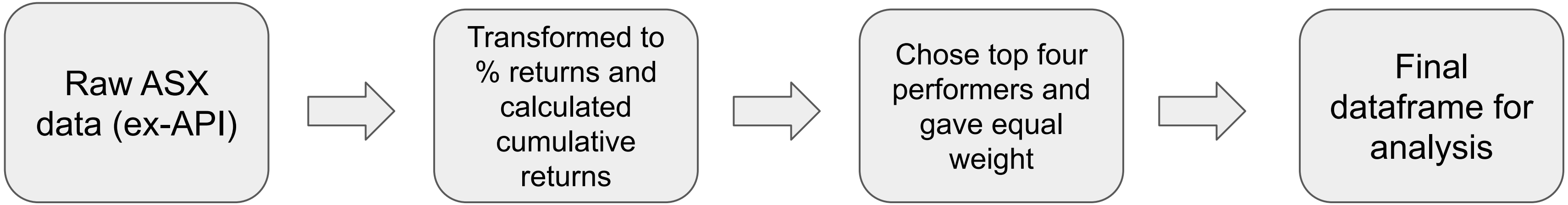
- Our study also provides a way to quantify the panic, and give investors the signal about the best entry time.



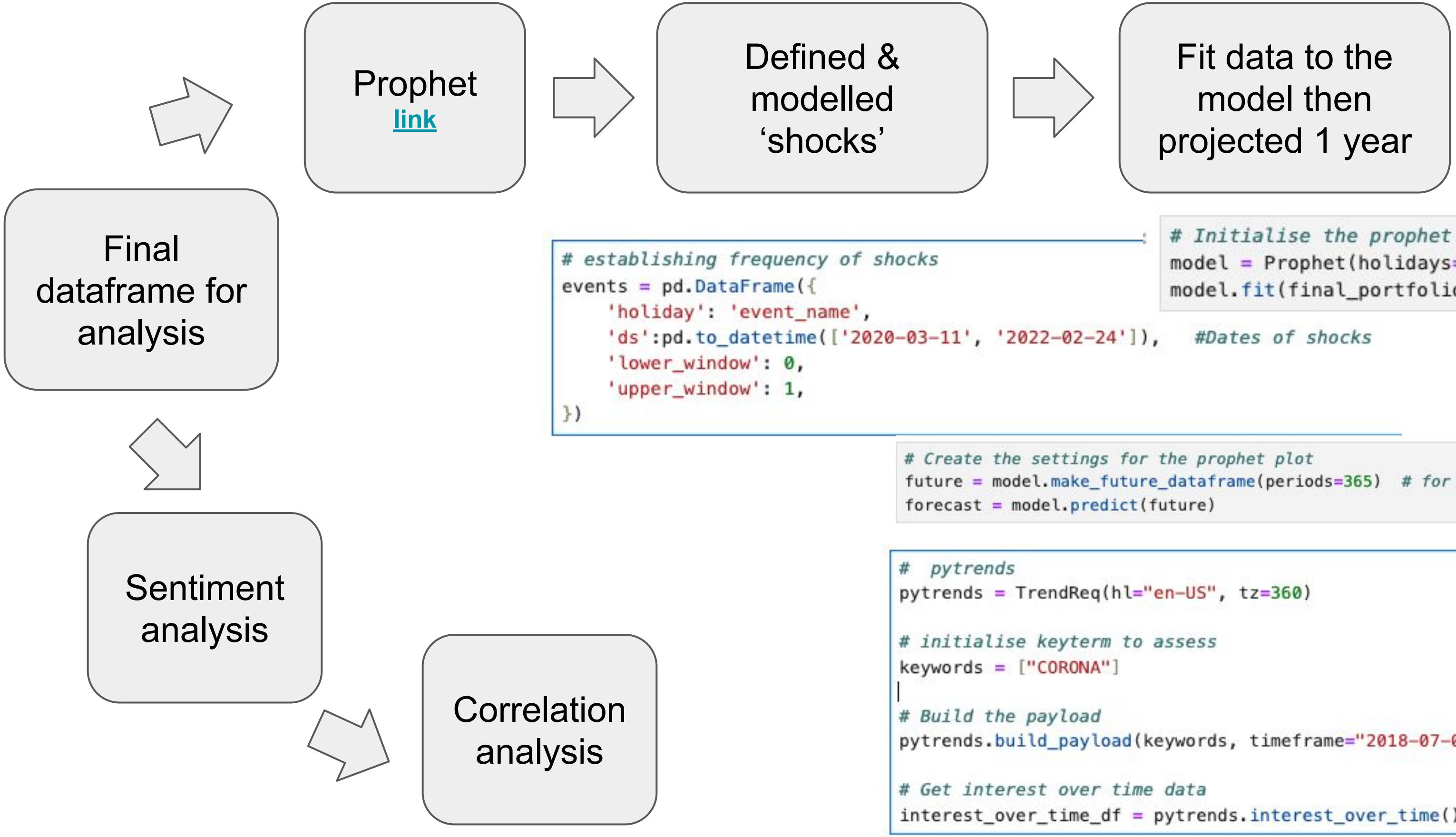

Source: Westcore Funds / Denver Investment Advisors LLC, 1998

# QUESTIONS & DATA

- Data we required:
    - ASX daily close data by sector
        - Sourced from Yahoo! Finance using the *yfinance* library API
    - Google trends data for ‘Corona’ and ‘Ukraine’
        - Sourced from Google Trends using the *pytrends* library API
        - Also tried ‘covid’ and ‘Putin’



# QUESTIONS & DATA (Flow of Transformed Data)




```
# Initialise the prophet model
model = Prophet(holidays=events)
model.fit(final_portfolio_df)
```

```
# establishing frequency of shocks
events = pd.DataFrame({
    'holiday': 'event_name',
    'ds':pd.to_datetime(['2020-03-11', '2022-02-24']),   #Dates of shocks
    'lower_window': 0,
    'upper_window': 1,
})
```

```
# Create the settings for the prophet plot
future = model.make_future_dataframe(periods=365)  # for a year into the future
forecast = model.predict(future)
```

```
#  pytrends
pytrends = TrendReq(hl="en-US", tz=360)

# initialise keyterm to assess
keywords = ["CORONA"]

# Build the payload
pytrends.build_payload(keywords, timeframe="2018-07-01 2023-06-30", geo="")

# Get interest over time data
interest_over_time_df = pytrends.interest_over_time()
```

# DATA CLEANUP & EXPLORATION

- For stock index data cleanup consistent of transforming daily close data to percentage changes and we identified the top performing indices using the cumulative returns
- Google Trends data is weekly, so for our sentiment analysis we had to bucket our daily returns to weekly.  This likely caused a loss of explanatory power for our model
- Using trends data need to be careful of the words you use: e.g. Covid vs Coronavirus, Ukraine vs Putin
- Using % changes with numbers like 0 or close to 0 can cause outsize effects on model, and need to be treated carefully.
- As shown later, using the Prophet library to model and predict shocks required reading of the Prophet library documentation, to ensure we had a model that could actually run

```
# Drop the Nan value from calculation of pct_change
ukraine_sentiment_change_drop = ukraine_sentiment_change.dropna()
ukraine_sentiment_change_drop
```

```
date
2022-01-09    0.000000
2022-01-16    2.000000
2022-01-23    1.000000
2022-01-30   -0.333333
2022-02-06    0.500000
2022-02-13    1.500000
```

```
# Replace the '0' values to be '1' from the UKRAINE sentiment df
ukraine_interest_over_time_df["UKRAINE"] = ukraine_interest_over_time_df["UKRAINE"].replace(0, 1)
ukraine_interest_over_time_df
```

| | UKRAINE | isPartial |
|---|---|---|
| date | | |
| 2022-01-02 | 1 | False |
| 2022-01-09 | 1 | False |
| 2022-01-16 | 3 | False |
| 2022-01-23 | 6 | False |
| 2022-01-30 | 4 | False |
| 2022-02-06 | 6 | False |

# DATA ANALYSIS

Jenny

```
# Calculate the percentage change of the data and attach it into a data frame
daily_returns = close_data.pct_change()
cumprod = (1+daily_returns).cumprod()
final_df = cumprod.iloc[-1]
final_df.sort_values()
```

```
^AXEJ    0.903417
^AXPJ    0.960164
^AXFJ    1.002727
^AXUJ    1.064989
^AXNJ    1.146523
^AXSJ    1.189669
^AXDJ    1.203074
^AXHJ    1.363937
^AXMJ    1.474104
^AXTJ    1.558498
^AXIJ    1.606659
Name: 2023-06-29 00:00:00, dtype: float64
```

- Use “.pct_change()” to calculate the daily returns
- Use “.cumprod()” to calculate cumulative portfolio daily returns

# DATA ANALYSIS (cont'd)

- Assign the weights to different sectors => determine the allocation of the initial investment across the portfolio
- Create a DataFrame for our investment portfolio

```
# Assign inital investment and weights into the notebook
weights = [.25, .25, .25, .25]
initial_investment = 100000
```

```
# Create a data frame for our own investment portfolio
investment_df = final_df.sort_values().iloc[-4:].index.values
investment_df
```

```
array(['^AXHJ', '^AXMJ', '^AXTJ', '^AXIJ'], dtype=object)
```

...

| Date | ^AXHJ | ^AXMJ | ^AXTJ | ^AXIJ |
|---|---|---|---|---|
| **2018-07-03** | 0.011239 | -0.014306 | 0.021410 | 0.003631 |
| **2018-07-04** | -0.003745 | -0.001530 | 0.005365 | -0.005735 |
| **2018-07-05** | 0.004357 | -0.002281 | 0.016996 | -0.003550 |
| **2018-07-06** | 0.004150 | 0.011240 | 0.013991 | 0.000534 |
| **2018-07-09** | -0.004369 | 0.010983 | 0.002396 | -0.000534 |
| ... | ... | ... | ... | ... |
| **2023-06-23** | -0.006572 | -0.012363 | -0.004174 | -0.011911 |
| **2023-06-26** | -0.005787 | -0.002352 | -0.000720 | 0.008798 |
| **2023-06-27** | -0.001200 | 0.011615 | -0.006488 | -0.003681 |
| **2023-06-28** | 0.004663 | 0.005938 | 0.010819 | 0.013074 |
| **2023-06-29** | 0.000614 | -0.007530 | 0.002349 | 0.018068 |

1264 rows × 4 columns

# DATA ANALYSIS (cont'd)

```
# assign the weights to each sector in 'investment_df'
final_portfolio = final_returns_df.dot(weights)
final_portfolio
```

=> Determine the overall performance of the investment portfolio based on the chosen allocation weight above.

```
# Reset the index in final portfolio
final_portfolio_df = final_portfolio.reset_index()
final_portfolio_df
```

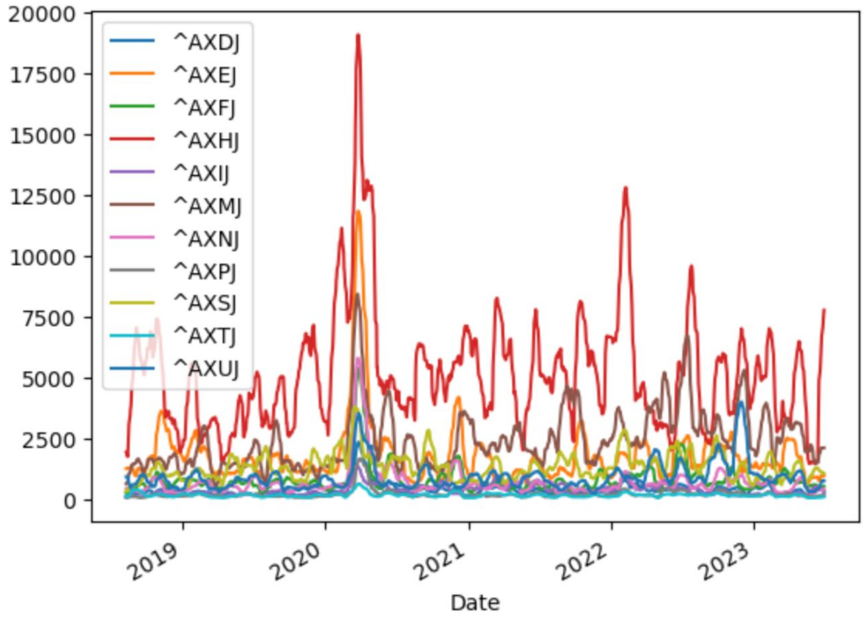
Use ".reset_index()" to reset the index of our DataFrame

# VOLATILITY

```
window_size = 30
volatility = close_data.rolling(window=window_size).std() * np.sqrt(window_size)
volatility.dropna()
volatility.plot()
```

Measure the volatility of stock prices in the past 5 years by using standard deviation.

<Axes: xlabel='Date'>




**Insights**

- AXHJ is the most volatile one. ⇒ Higher risk
- The start of 2020 is the most volatile period for all stocks. ⇒ COVID

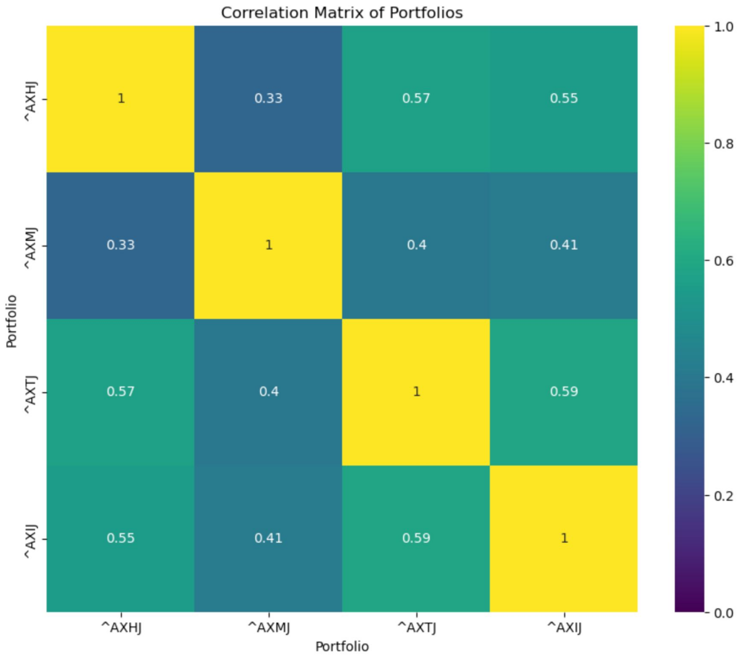
```python
# Display de correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='viridis', vmin=0, vmax=1)

#Set plotting title and labels
plt.title('Correlation Matrix of Portfolios')
plt.xlabel('Portfolio')
plt.ylabel('Portfolio')
plt.show()
```


```python
final_correlation_matrix = final_returns_df.corr()

# Display de correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(final_correlation_matrix, annot=True, cmap='viridis', vmin=0, vmax=1)

#Set plotting title and labels
plt.title('Correlation Matrix of Portfolios')
plt.xlabel('Portfolio')
plt.ylabel('Portfolio')
plt.show()
```



DD

# FORECAST RESULTS BY USING "PROPHET" MODEL

```
# Plot the prophet model
prophet_plot = (model.plot(forecast),
    plt.legend(),
    plt.title("Cumulative Returns Forecast for adjusted Sector"),
    plt.xlabel("Date"),
    plt.ylabel("Cumulative Returns")
       )
```

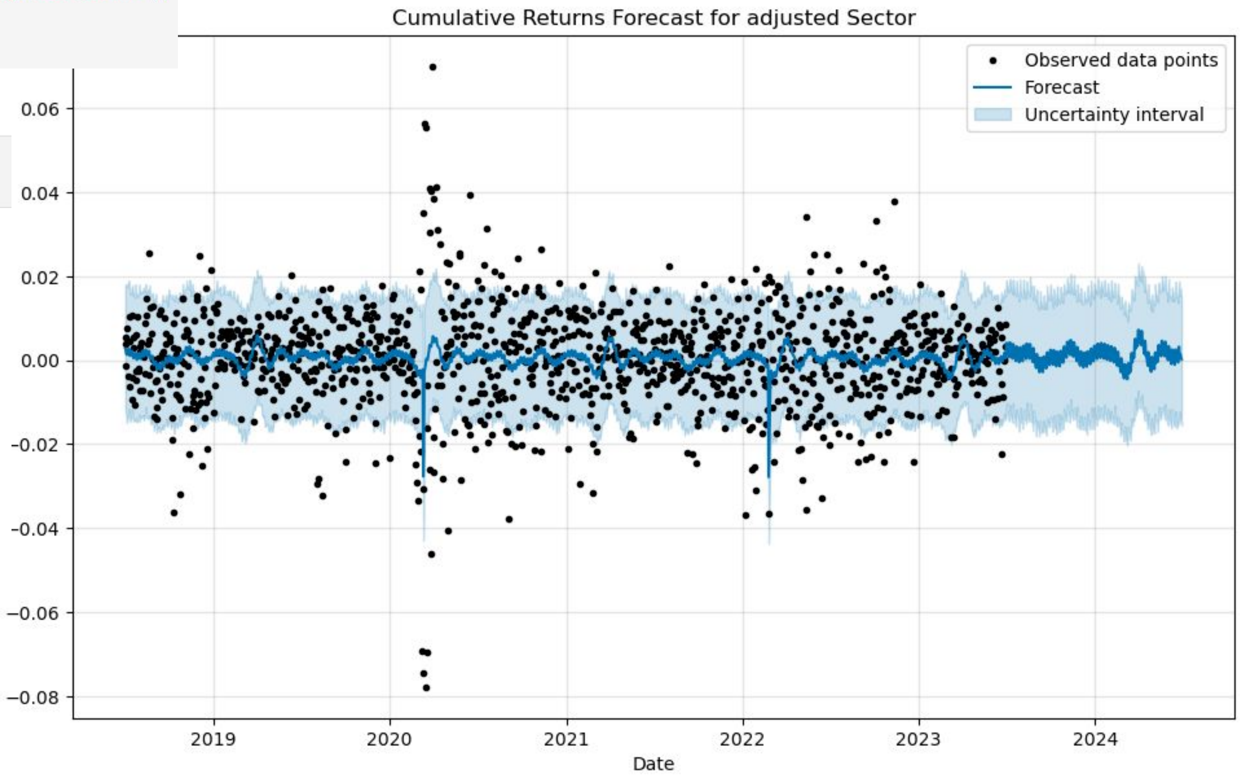
In [9]:

```
# Rename the columns to accommodate prophet library
final_portfolio_df.columns = ['ds','y']
final_portfolio_df
```

Out[9]:

| | ds | y |
|---|---|---|
| 0 | 2018-07-03 | 0.005494 |
| 1 | 2018-07-04 | -0.001411 |
| 2 | 2018-07-05 | 0.003881 |
| 3 | 2018-07-06 | 0.007479 |
| 4 | 2018-07-09 | 0.002119 |
| ... | ... | ... |
| 1259 | 2023-06-23 | -0.008755 |
| 1260 | 2023-06-26 | -0.000015 |
| 1261 | 2023-06-27 | 0.000061 |
| 1262 | 2023-06-28 | 0.008623 |
| 1263 | 2023-06-29 | 0.003375 |

1264 rows × 2 columns

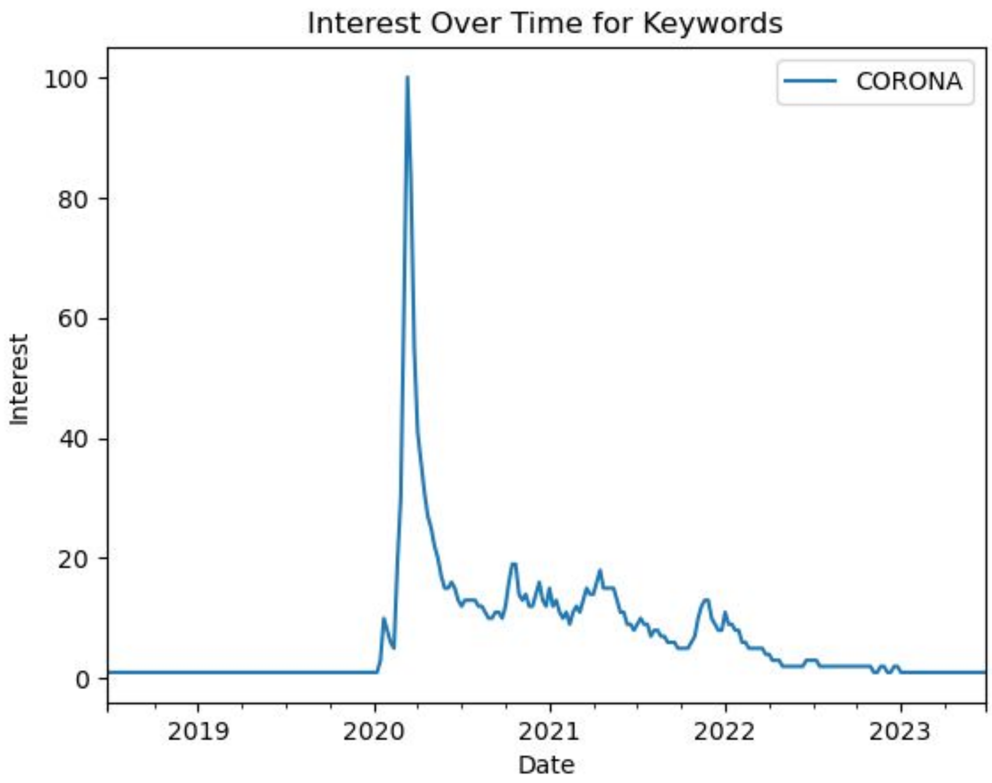


# Presence of the term ‘CORONA’

Visualise the popularity of “CORONA” from its appearance on google content

**Observations:**

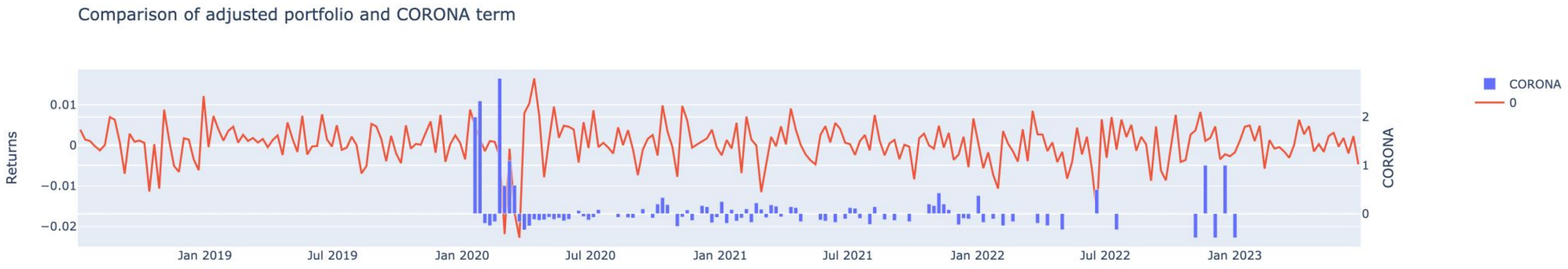
- Highs and lows entering and exiting lockdown
- Waves influenced global search popularity



# “CORONA” impact on returns

## Insights

- In the beginning of COVID, people’s attention to pandemic was increasing
- Interest climbed again toward the ‘end’ of the pandemic
- The period of highest sentiment change overlaps with the most volatile time in the market



# Presence of the term ‘UKRAINE’

**Insights:**

Compared to CORONA, the interest over Ukraine War doesn’t have much impact on the portfolio return.

```
correlation_plot = make_subplots(rows=1, cols=1, shared_xaxes=True, specs=[[{'secondary_y': True}]])

#Add bar plot plot for the UKRAINE term frequencies
correlation_plot.add_trace(go.Bar(x=ukraine_combined_data['index'], y=ukraine_combined_data['UKRAINE'], name='UKRAINE'), secondary_y=True)

correlation_plot.add_trace(go.Scatter(x=ukraine_combined_data['index'], y=ukraine_combined_data[0], mode='lines', name=0), secondary_y=False)

correlation_plot.update_layout(title='Comparison of adjusted portfolio and UKRAINE term')

correlation_plot.update_yaxes(title_text='UKRAINE', secondary_y=True)
correlation_plot.update_yaxes(title_text='Returns', secondary_y=False)

correlation_plot.show()
```

# DISCUSSION OF FINDINGS

| **Expectations** | **Findings** |
| --- | --- |
| ● High correlation between returns and sentiment<br>● Market shocks harsher<br>● Global impact from Ukraine<br>● All sectors performance impacted | ● Almost no relation between sectors and sentiment<br>● Market resilience<br>● Temporal trends in the market<br>● Ukraine conflict minimal impact<br>● ‘CORONA’ provided better data than ‘COVID’ |

# Postmortem

- Analysis of markets or equities closer related to shocks
- Analyse longer period of time with more shocks (more data looking backwards)
- Expand the analysis across other asset classes
- Investment into a better API - Twitter API with granular data
- Correlation between two markets geographically apart