Index crashes

October 19, 2022

[4]:
```
import numpy as np
import pandas as pd
from pandas_datareader import data as wb
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as st
```

We define a “index crash” as a huge daily change in price (>5 standard deviations from the mean over the last trading year).

We’re applying the same method illustrated in Mandelbrot’s “The Misbehaviour Of Markets” (2008, Mathematical appendix).

The goal is to show how unlikely a high price change is according to a normal distribution. This in turn provides evidence of the uselessness of using a normal distribution to quantify risk.

First some code to import prices for individual indexes from Yahoo.

[5]:
```
def import_asset_data(tickers, start = '2010-1-1'):
    """
    Imports data of a list of assets from a certain date.

    Parameters
    ----------
    tickers : list of str
        List of tickers for asset prices
    start : str
        Date in yyyy-MM-dd format

    Returns
    -------
    pandas.DataFrame
    """
    data = pd.DataFrame()
    if len([tickers]) ==1:
        data[tickers] = wb.DataReader(tickers, data_source='yahoo', start =␣
↪start)['Adj Close']
        #data = pd.DataFrame(data)
    else:
```

```python
        for t in tickers:
            data[t] = wb.DataReader(t, data_source='yahoo', start = start)['Adj Close']
    return(data)
```

Import index data from Yahoo. We're considering the top 6 indexes according to https://finance.yahoo.com/world-indices/?guccounter=1&guce_referrer=aHR0cHM6Ly9kdWNrZHVja2dvLmNvbS8&guce_referrer_sig=AQAAA W0fEaaUFSzEELd-X9fRbZFjj2gF2hXHZJEvUqHPMzg at the time of writing (19/10/2022).

[11]:
```python
asset = ['^GSPC', '^DJI', '^IXIC', '^NYA', '^XAX', '^BUK100P']
data = import_asset_data(asset)
```

[12]:
```python
data.head()
```

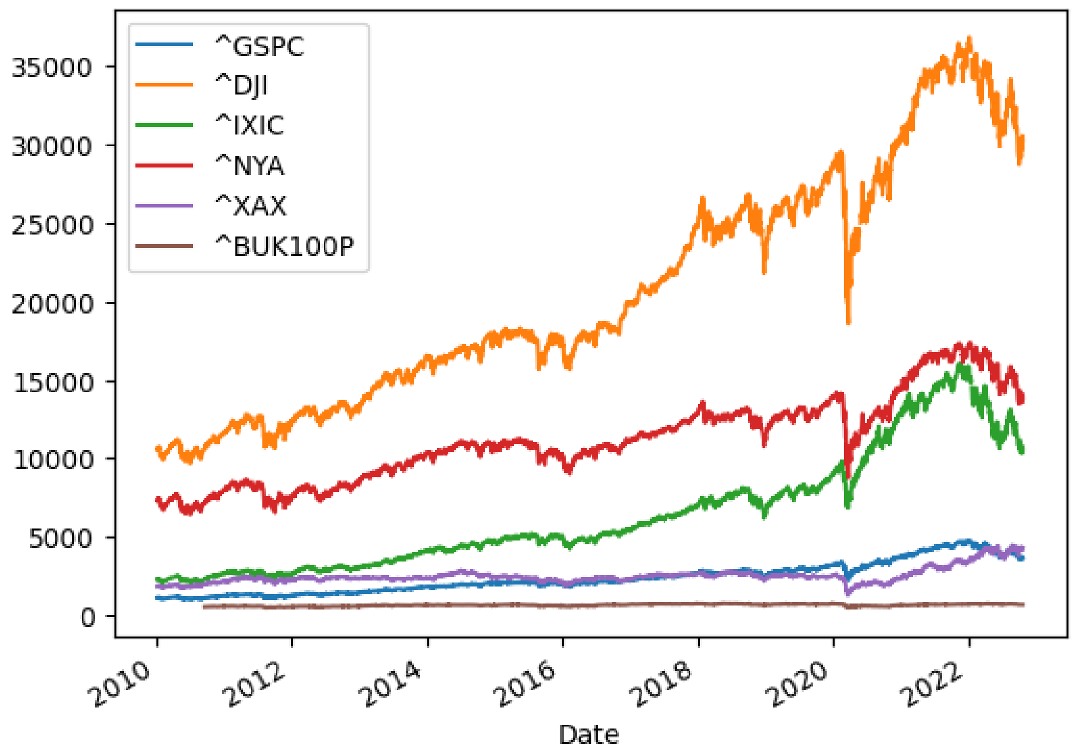
[12]:
```
                  ^GSPC          ^DJI        ^IXIC         ^NYA         ^XAX  \
Date
2010-01-04  1132.989990  10583.959961  2308.419922  7326.740234  1853.660034
2010-01-05  1136.520020  10572.019531  2308.709961  7354.870117  1859.920044
2010-01-06  1137.140015  10573.679688  2301.090088  7377.700195  1866.900024
2010-01-07  1141.689941  10606.860352  2300.050049  7393.930176  1868.020020
2010-01-08  1144.979980  10618.190430  2317.169922  7425.350098  1872.500000

            ^BUK100P
Date
2010-01-04       NaN
2010-01-05       NaN
2010-01-06       NaN
2010-01-07       NaN
2010-01-08       NaN
```

Plot stocks price.

[13]:
```python
data[asset].plot()
```

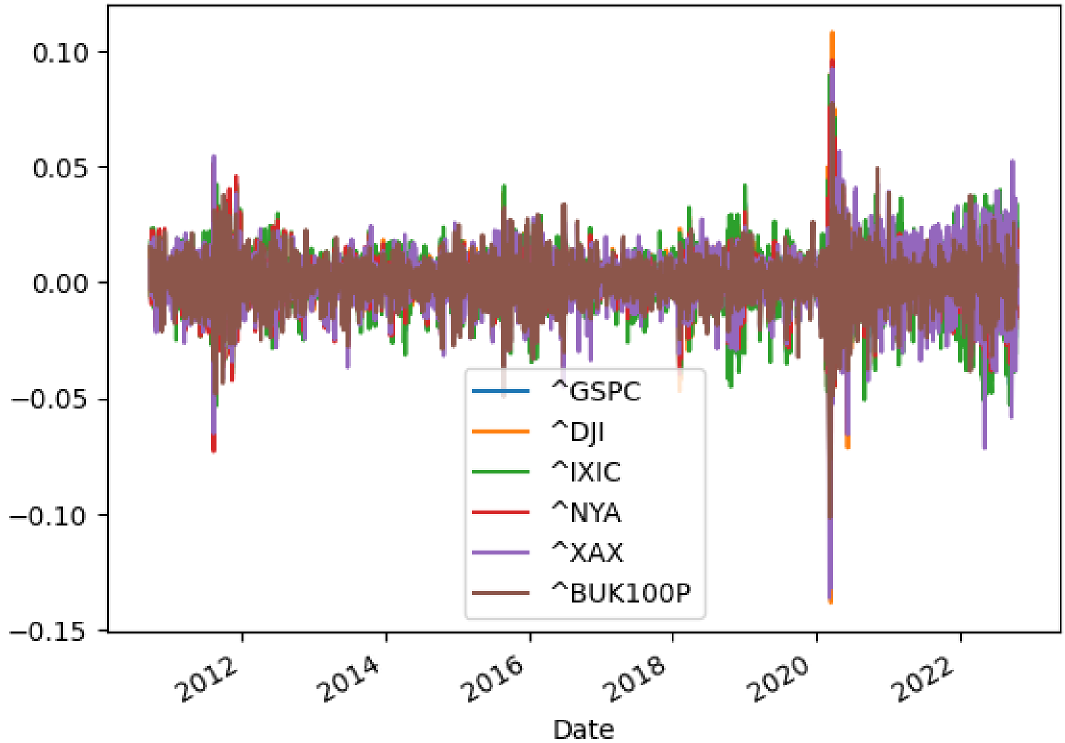
[13]: <AxesSubplot: xlabel='Date'>



Compute logarithm of prices and plot daily changes. We're interested in modeling price changes, not price itself.

[22]:
```
log_data = np.log(data[asset])
log_data_diff = log_data.diff().dropna()
log_data_diff.plot()
```

[22]: <AxesSubplot: xlabel='Date'>



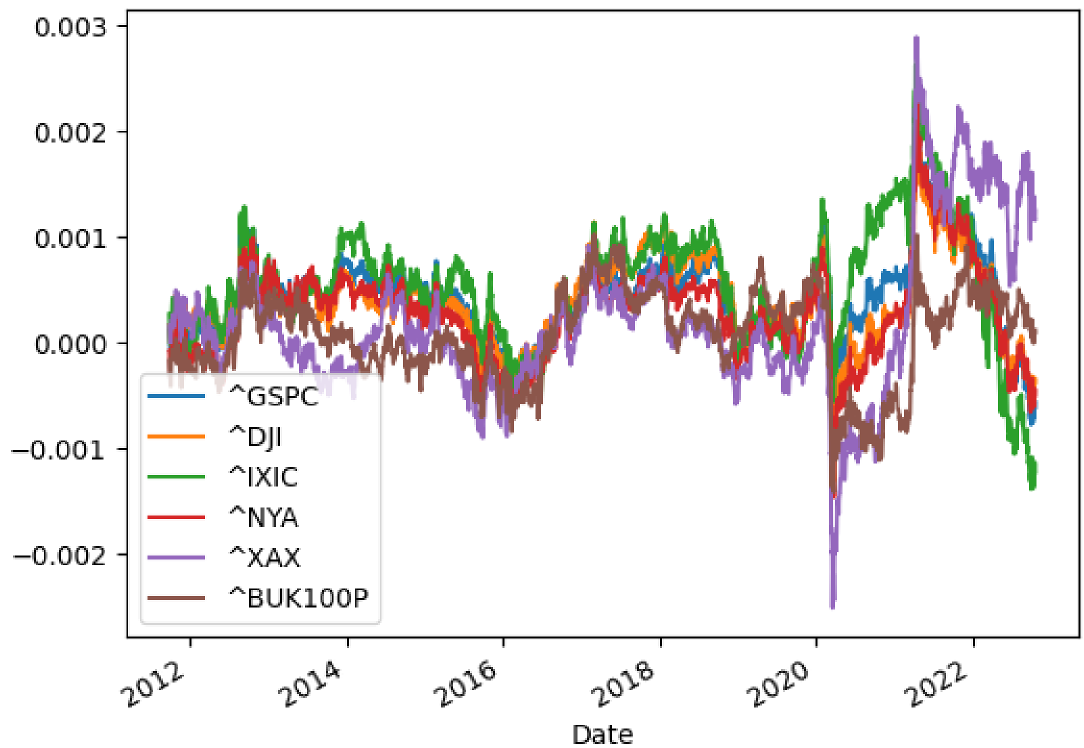
Define window length to be used in all subsequent computations. We consider mean, volatility, etc., over the course of one trading year (250 days).

[16]:
```
window_length = 250
```

Compute mean.

[23]:
```
mean = log_data_diff.rolling(window_length).mean()
mean.plot()
```

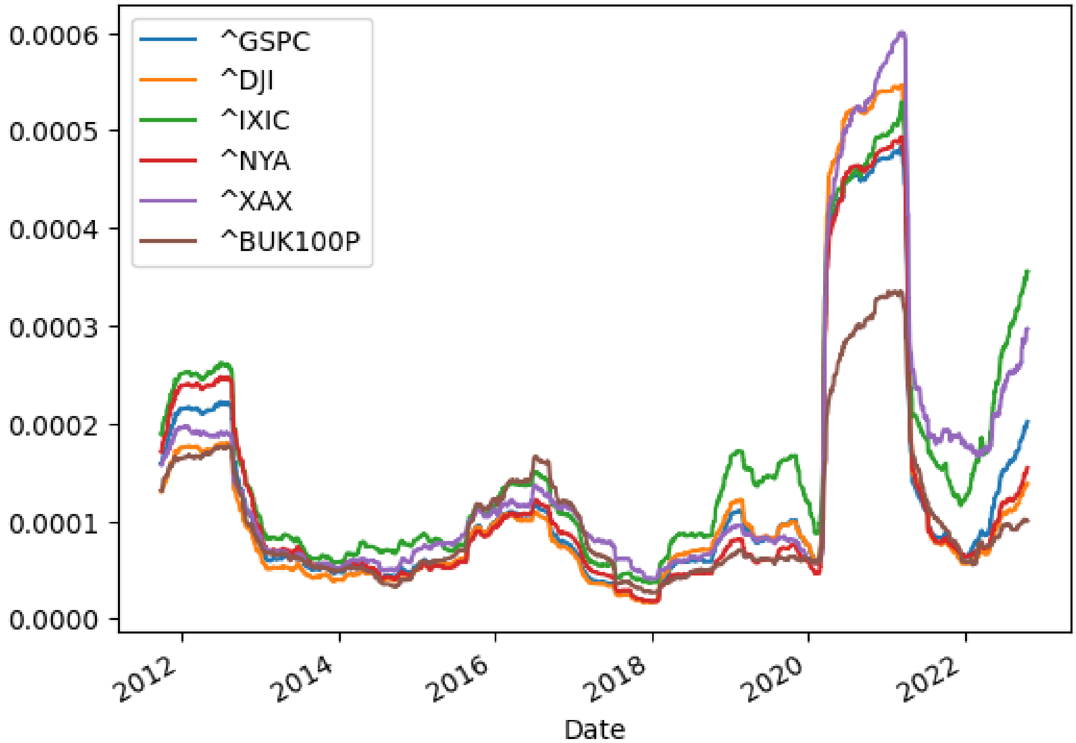
[23]: <AxesSubplot: xlabel='Date'>



Compute variance (as if it would fit the bell curve).

[24]:
```
var = log_data_diff.rolling(window_length).var()
var.plot()
```

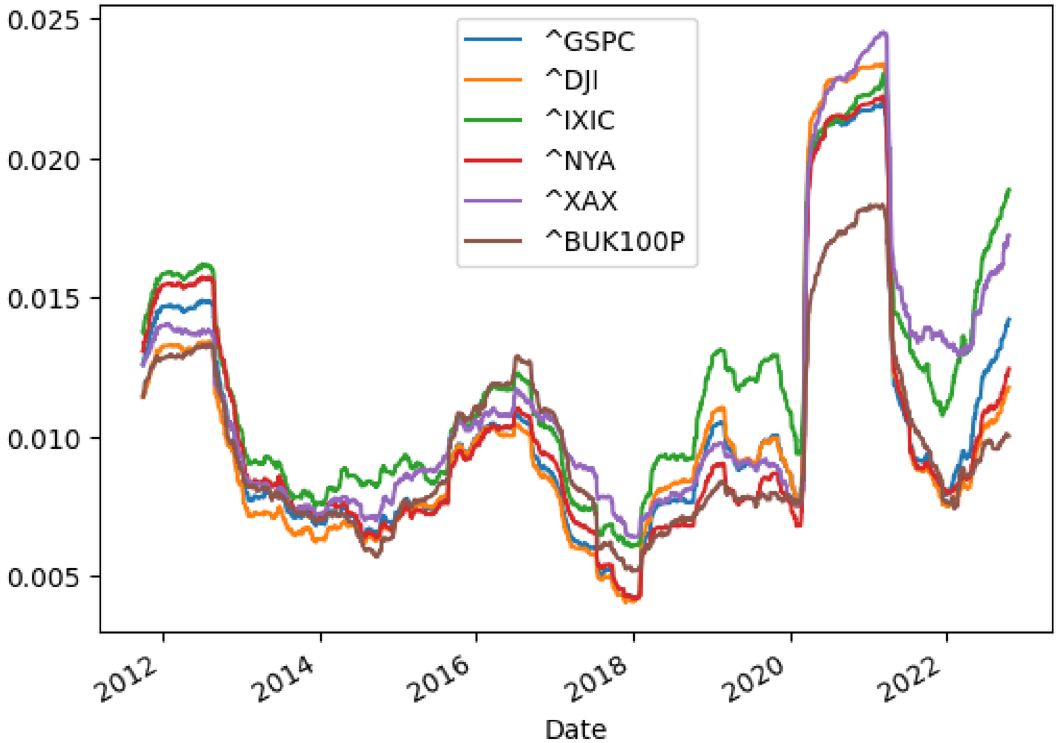
[24]: <AxesSubplot: xlabel='Date'>



Compute standard deviation.

[25]:
```
std = log_data_diff.rolling(window_length).std()
std.plot()
```

[25]: <AxesSubplot: xlabel='Date'>

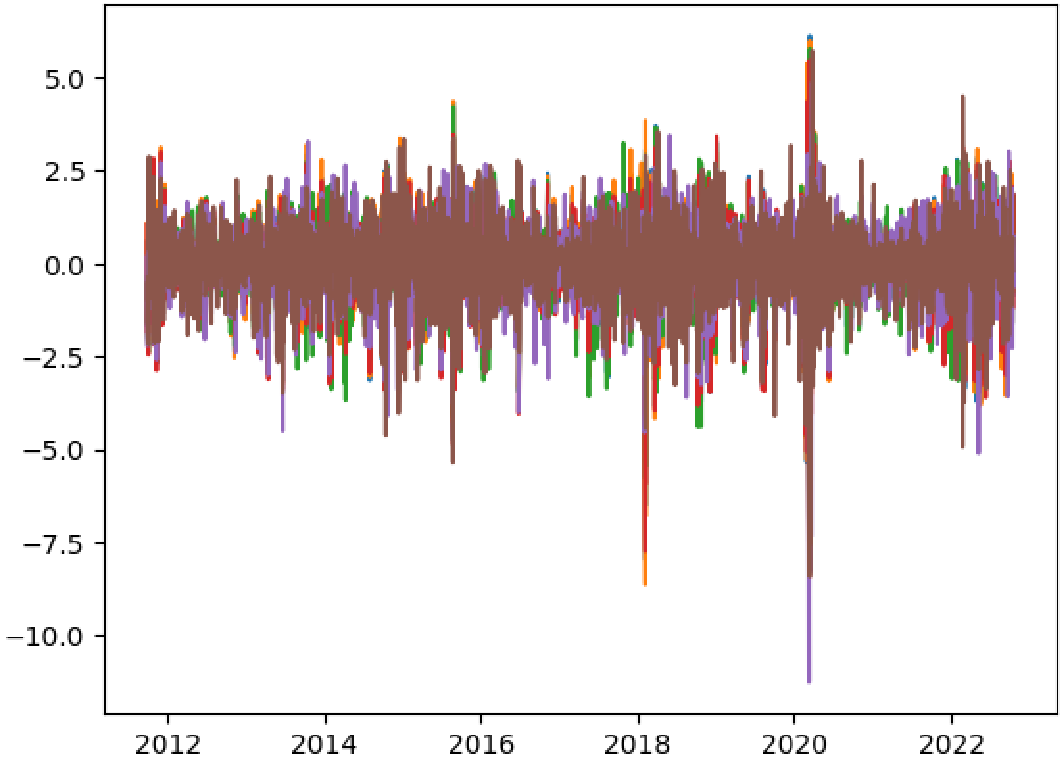


As we can see from this plots, variance varies wildly.

Find number of standard deviations from the mean for each day (z-score). We start after the first year, minus one because we're dealing with price changes. z-score measures how much each day is distant from the mean, expressed in standard deviations over the previous year.

[26]:
```
zscores = (log_data_diff[window_length-1:] - mean) / std
#zscores = st.zscore(log_data_diff.dropna().to_numpy())
plt.plot(zscores)
```

[26]:
```
[<matplotlib.lines.Line2D at 0x7fc721af1e40>,
 <matplotlib.lines.Line2D at 0x7fc721af2080>,
 <matplotlib.lines.Line2D at 0x7fc721af20b0>,
 <matplotlib.lines.Line2D at 0x7fc721af1ff0>,
 <matplotlib.lines.Line2D at 0x7fc721af1f00>,
 <matplotlib.lines.Line2D at 0x7fc721af21a0>]
```

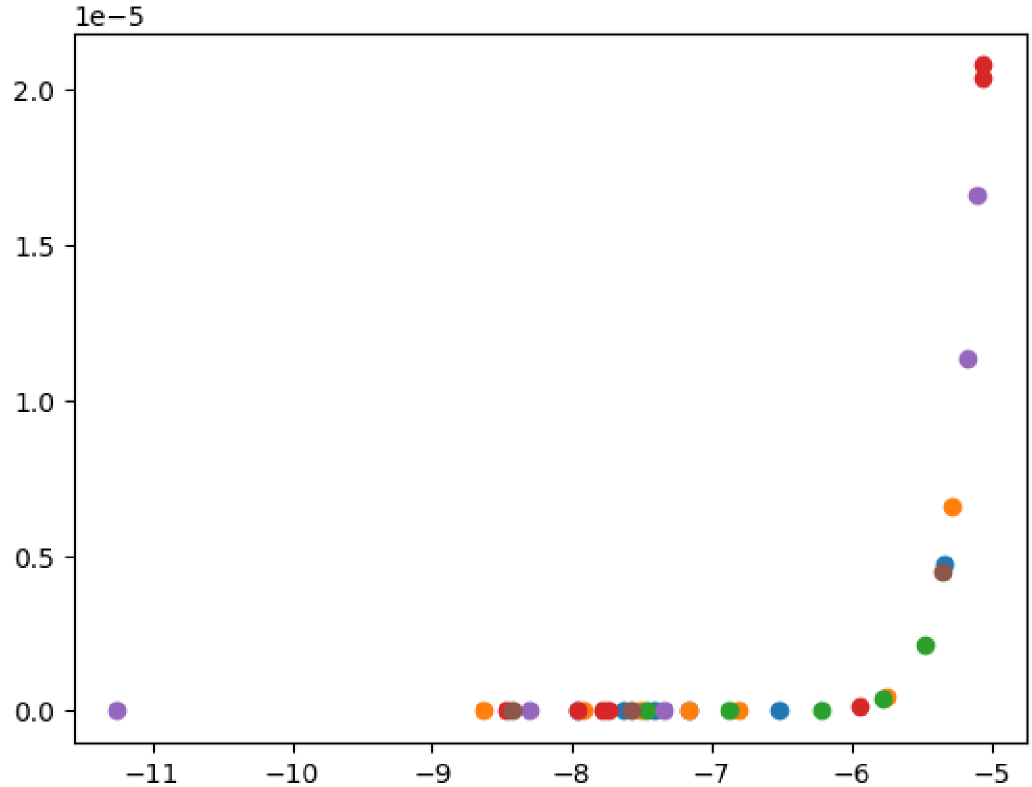


Find crashes (events with absolute high z-score, but in the negative).

[27]:
```
crashes = pd.DataFrame()
crashes[asset] = zscores[asset][zscores < -5]
```

Find probability of each crash. That is, getting a daily change as extreme as those in the data. We're considering those at least 5 standard deviations apart.

[28]:
```
zscores_prob = pd.DataFrame()
zscores_prob[asset] = st.norm(0, 1).cdf(crashes)
for a in asset:
    plt.scatter(crashes[a], zscores_prob[a]*100) # percentage
```



If price changes really did follow a normal distributions, such crashes would be almost impossible (even considering the most probable, that's still a 0.00002% probability of such a crash).