Choose your time series

In [40]:

```
import pandas as pd

pd.set_option('display.max_columns', 200)
pd.set_option('display.max_rows', 200)

df_src = pd.read_csv('SP500_1979.csv', sep=',')
print(f'Lines and columns: {df_src.shape}\n')
df_src.head()
```

```
Lines and columns: (10087, 7)
```

Out[40]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1979-04-16	102.000000	102.019997	100.669998	101.120003	101.120003	28050000
1	1979-04-17	101.120003	101.940002	100.650002	101.239998	101.239998	29260000
2	1979-04-18	101.239998	102.230003	100.959999	101.699997	101.699997	29510000
3	1979-04-19	101.699997	102.400002	100.879997	101.279999	101.279999	31150000
4	1979-04-20	101.279999	101.809998	100.459999	101.230003	101.230003	28830000

In [41]:

```
data = []
df_src_len = len(df_src['Adj Close'])

for i in range(df_src_len):
    if i < df_src_len - 1:
        data.append(df_src['Adj Close'][i] / df_src['Adj Close'][i + 1] - 1)
    else:
        data.append(0)

df_src['Returns'] = data
df_src.head()
```

Out[41]:

	Date	Open	High	Low	Close	Adj Close	Volume	Returns
0	1979-04-16	102.000000	102.019997	100.669998	101.120003	101.120003	28050000	-0.001185
1	1979-04-17	101.120003	101.940002	100.650002	101.239998	101.239998	29260000	-0.004523
2	1979-04-18	101.239998	102.230003	100.959999	101.699997	101.699997	29510000	0.004147
3	1979-04-19	101.699997	102.400002	100.879997	101.279999	101.279999	31150000	0.000494
4	1979-04-20	101.279999	101.809998	100.459999	101.230003	101.230003	28830000	-0.003347

Choose your ranges

In [42]:

```
ranges = {1: [], 2: [], 4: [], 8: [], 16: [], 32: []}

for r in ranges:
    piece = df_src.shape[0]/r
    interval = 0
    for i in range(r):
        end = interval + piece
        ranges[r].append({'start': interval, 'end': end})
        interval += piece

print(ranges)
```

```
{1: [{'start': 0, 'end': 10087.0}], 2: [{'start': 0, 'end': 5043.5},
{'start': 5043.5, 'end': 10087.0}], 4: [{'start': 0, 'end': 2521.7
5}, {'start': 2521.75, 'end': 5043.5}, {'start': 5043.5, 'end': 756
5.25}, {'start': 7565.25, 'end': 10087.0}], 8: [{'start': 0, 'end':
1260.875}, {'start': 1260.875, 'end': 2521.75}, {'start': 2521.75,
'end': 3782.625}, {'start': 3782.625, 'end': 5043.5}, {'start': 504
3.5, 'end': 6304.375}, {'start': 6304.375, 'end': 7565.25}, {'star
t': 7565.25, 'end': 8826.125}, {'start': 8826.125, 'end': 10087.0}],
16: [{'start': 0, 'end': 630.4375}, {'start': 630.4375, 'end': 1260.
875}, {'start': 1260.875, 'end': 1891.3125}, {'start': 1891.3125, 'e
nd': 2521.75}, {'start': 2521.75, 'end': 3152.1875}, {'start': 3152.
1875, 'end': 3782.625}, {'start': 3782.625, 'end': 4413.0625}, {'sta
rt': 4413.0625, 'end': 5043.5}, {'start': 5043.5, 'end': 5673.9375},
{'start': 5673.9375, 'end': 6304.375}, {'start': 6304.375, 'end': 69
34.8125}, {'start': 6934.8125, 'end': 7565.25}, {'start': 7565.25,
'end': 8195.6875}, {'start': 8195.6875, 'end': 8826.125}, {'start':
8826.125, 'end': 9456.5625}, {'start': 9456.5625, 'end': 10087.0}],
32: [{'start': 0, 'end': 315.21875}, {'start': 315.21875, 'end': 63
0.4375}, {'start': 630.4375, 'end': 945.65625}, {'start': 945.65625,
'end': 1260.875}, {'start': 1260.875, 'end': 1576.09375}, {'start':
1576.09375, 'end': 1891.3125}, {'start': 1891.3125, 'end': 2206.5312
5}, {'start': 2206.53125, 'end': 2521.75}, {'start': 2521.75, 'end':
2836.96875}, {'start': 2836.96875, 'end': 3152.1875}, {'start': 315
2.1875, 'end': 3467.40625}, {'start': 3467.40625, 'end': 3782.625},
{'start': 3782.625, 'end': 4097.84375}, {'start': 4097.84375, 'end':
4413.0625}, {'start': 4413.0625, 'end': 4728.28125}, {'start': 4728.
28125, 'end': 5043.5}, {'start': 5043.5, 'end': 5358.71875}, {'star
t': 5358.71875, 'end': 5673.9375}, {'start': 5673.9375, 'end': 5989.
15625}, {'start': 5989.15625, 'end': 6304.375}, {'start': 6304.375,
'end': 6619.59375}, {'start': 6619.59375, 'end': 6934.8125}, {'star
t': 6934.8125, 'end': 7250.03125}, {'start': 7250.03125, 'end': 756
5.25}, {'start': 7565.25, 'end': 7880.46875}, {'start': 7880.46875,
'end': 8195.6875}, {'start': 8195.6875, 'end': 8510.90625}, {'star
t': 8510.90625, 'end': 8826.125}, {'start': 8826.125, 'end': 9141.34
375}, {'start': 9141.34375, 'end': 9456.5625}, {'start': 9456.5625,
'end': 9771.78125}, {'start': 9771.78125, 'end': 10087.0}]}
```

Calculate the mean for each range

In [43]:

```
data = []

for r in ranges:
    for i in range(r):
        start = ranges[r][i]['start']
        end = ranges[r][i]['end']
        mean = df_src.loc[start:end]['Returns'].mean()
        data.append([r, i+1, start, end, mean])

df_means = pd.DataFrame(data, columns = ['range', 'index', 'start', 'end', 'mea
n'])
df_means.head()
```

Out[43]:

	range	index	start	end	mean
0	1	1	0.00	10087.00	-0.000272
1	2	1	0.00	5043.50	-0.000457
2	2	2	5043.50	10087.00	-0.000088
3	4	1	0.00	2521.75	-0.000365
4	4	2	2521.75	5043.50	-0.000550

Create a serie of deviations for each range

In [44]:

```
data = []
df_deviation = pd.DataFrame([])

for r in ranges:
    for i in range(r):
        start = ranges[r][i]['start']
        end = ranges[r][i]['end']
        serieMean = df_means[df_means['range'] == r][i:i+1]['mean']
        for j in range(int(start), int(end)):
            diff = df_src['Returns'][j] - float(serieMean)
            data.append(diff)
    df_deviation[r] = data
    data = []

df_deviation.head()
```

Out[44]:

	1	2	4	8	16	32
0	-0.000913	-0.000728	-0.000821	-0.000888	-0.000944	-0.000699
1	-0.004251	-0.004066	-0.004159	-0.004226	-0.004282	-0.004037
2	0.004419	0.004604	0.004511	0.004444	0.004388	0.004633
3	0.000766	0.000951	0.000858	0.000791	0.000735	0.000980
4	-0.003075	-0.002890	-0.002983	-0.003050	-0.003106	-0.002862

Create a series which is running total of the deviations from the mean

In [46]:

```
df_deviation_total = df_deviation.copy()

for c in df_deviation.columns:
    data = []
    prevValue = 0
    for v in df_deviation[c]:
        data.append(v + prevValue)
        prevValue = v + prevValue
    df_deviation_total[c] = data

df_deviation_total.head()
```

Out[46]:

	1	2	4	8	16	32
0	-0.000913	-0.000728	-0.000821	-0.000888	-0.000944	-0.000699
1	-0.005163	-0.004794	-0.004979	-0.005113	-0.005225	-0.004737
2	-0.000744	-0.000190	-0.000468	-0.000669	-0.000837	-0.000104
3	0.000022	0.000761	0.000391	0.000123	-0.000102	0.000876
4	-0.003053	-0.002129	-0.002592	-0.002927	-0.003207	-0.001986

Calculate the widest difference in the series of deviations

In [47]:

```
max_values, min_values, diff_values = [], [], []

for r in ranges:
    for i in range(r):
        start = ranges[r][i]['start']
        end = ranges[r][i]['end']
        max_v = max(df_deviation_total.loc[start:end][r])
        min_v = min(df_deviation_total.loc[start:end][r])
        diff = max_v - min_v
        max_values.append(max_v)
        min_values.append(min_v)
        diff_values.append(diff)

df_means['max'] = max_values
df_means['min'] = min_values
df_means['diff_range'] = diff_values
df_means.head()
```

Out[47]:

	range	index	start	end	mean	max	min	diff_range
0	1	1	0.00	10087.00	-0.000272	0.637641	-1.006057	1.643699
1	2	1	0.00	5043.50	-0.000457	0.522299	-0.151942	0.674241
2	2	2	5043.50	10087.00	-0.000088	1.107191	-0.120546	1.227738
3	4	1	0.00	2521.75	-0.000365	0.326785	-0.347874	0.674658
4	4	2	2521.75	5043.50	-0.000550	0.421831	-0.123210	0.545041

Calculate the standard deviation for each range

In [48]:

```
standard_deviations = []

for r in ranges:
    for i in range(r):
        start = ranges[r][i]['start']
        end = ranges[r][i]['end']
        standard_deviation = df_deviation.loc[start:end][r].std()
        standard_deviations.append(standard_deviation)

df_means['standard_deviation'] = standard_deviations
df_means.head()
```

Out[48]:

	range	index	start	end	mean	max	min	diff_range	standard_dev
0	1	1	0.00	10087.00	-0.000272	0.637641	-1.006057	1.643699	0.0
1	2	1	0.00	5043.50	-0.000457	0.522299	-0.151942	0.674241	0.0
2	2	2	5043.50	10087.00	-0.000088	1.107191	-0.120546	1.227738	0.0
3	4	1	0.00	2521.75	-0.000365	0.326785	-0.347874	0.674658	0.0
4	4	2	2521.75	5043.50	-0.000550	0.421831	-0.123210	0.545041	0.0

Calculate the rescaled range for each range in the time series

In [49]:

```
df_means['rescaled_range'] = df_means['diff_range'] / df_means['standard_deviation']
df_means.head()
```

Out[49]:

	range	index	start	end	mean	max	min	diff_range	standard_dev
0	1	1	0.00	10087.00	-0.000272	0.637641	-1.006057	1.643699	0.0
1	2	1	0.00	5043.50	-0.000457	0.522299	-0.151942	0.674241	0.0
2	2	2	5043.50	10087.00	-0.000088	1.107191	-0.120546	1.227738	0.0
3	4	1	0.00	2521.75	-0.000365	0.326785	-0.347874	0.674658	0.0
4	4	2	2521.75	5043.50	-0.000550	0.421831	-0.123210	0.545041	0.0

Average the rescaled range values for each region to summarize each range

In [50]:

```
data = []
columns = ['range', 'average_rescaled', 'size']

for r in ranges:
    size = df_means[df_means['range'] == r]['end'].iloc[0]
    mean = df_means[df_means['range'] == r]['rescaled_range'].mean()
    data.append([r, mean, size])

df_average_rescaled = pd.DataFrame(data, columns=columns)
df_average_rescaled
```

Out[50]:

	range	average_rescaled	size
0	1	148.723400	10087.00000
1	2	84.824875	5043.50000
2	4	59.751002	2521.75000
3	8	40.328881	1260.87500
4	16	28.842452	630.43750
5	32	19.136974	315.21875

Calculating the Hurst Expoent Steps

Calculate the logarithmic for the size of each region and for each region's rescaled range

In [51]:

```
import numpy as np

df_average_rescaled['log_RS'] = np.log10(df_average_rescaled['average_rescaled'
])
df_average_rescaled['log_size'] = np.log10(df_average_rescaled['size'])
df_average_rescaled
```

Out[51]:

	range	average_rescaled	size	log_RS	log_size
0	1	148.723400	10087.00000	2.172379	4.003762
1	2	84.824875	5043.50000	1.928523	3.702732
2	4	59.751002	2521.75000	1.776345	3.401702
3	8	40.328881	1260.87500	1.605616	3.100672
4	16	28.842452	630.43750	1.460032	2.799642
5	32	19.136974	315.21875	1.281873	2.498612

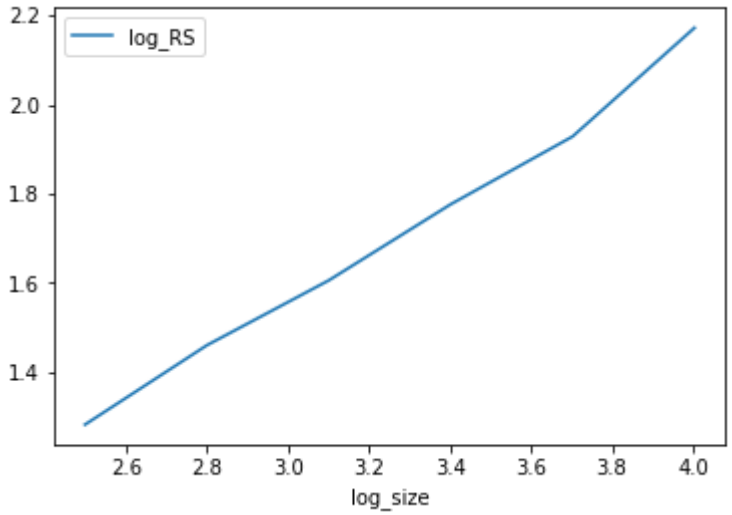
Plot the logarithm of the size (x axis) of each versus the logarithm of the rescaled range (y axis)

In [52]:

```
df_average_rescaled[['log_size', 'log_RS']].plot(x = 'log_size', y = 'log_RS')
```

Out[52]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f56d39a6b70>
```



Calculate the slope of the data to find the Hurst exponent

In [53]:

```
slopes = []

for i in range(len(ranges)):
    if i < len(ranges) - 1:
        y2 = df_average_rescaled['log_RS'][i]
        y1 = df_average_rescaled['log_RS'][i + 1]
        x2 = df_average_rescaled['log_size'][i]
        x1 = df_average_rescaled['log_size'][i + 1]
        slopes.append((y2 - y1) / (x2 - x1))

H = np.median(slopes)
H
```

Out[53]:

```
0.5671495455476077
```

Extra - Fractal dimension

In [54]:

```
fractalDimension = 2 - H
fractalDimension
```

Out[54]:

```
1.4328504544523923
```

In [ ]:

```

```