Samsung Innovation Campus

Coding, Programming & Data Science

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 8.

Data Analysis and Visualization Mini Project

Coding, Programming & Data Science

Chapter Description

Chapter objectives

- ✓ Be able to be confident by making the same results with the data analysis and visualization as experts make in the field only with what we have learned through the course, which is the core goal of the mini-project.
- ✓ Be able to understand that data analysis skills using Python are essential skills in the real field by solving real-world cases through the mini-project.
- ✓ Be able to practice in the field of financial accounting through financial data analysis tasks.
- ✓ Be able to practice on solving various real-world social problems through public data analysis tasks.

Chapter contents

- ✓ Unit 39. Financial Data Analysis Mini Project
- ✓ Unit 40. Global Corona Pandemic Analysis Mini Project

Unit 39.

Financial Data Analysis Mini Project

Mission

Financial Data Analysis Mini Project

- With the rapid increase in computing speed, mankind has been able to organize and analyze large amounts of data that have never been experienced before at incredibly high speed.
- Based on that, it is possible for computers to be learned and be developed to the current status of artificial intelligence.
- Even if you do not become a software engineer, you can solve many practical problems that you may see in the real world through the data processing skills you have learned so far using Python and especially Pandas.
- Pandas, a core library of data analysis applied in various fields, was originally developed to analyze and organize financial data.
- The core of data analysis is financial data analysis. Through the practice of acquiring and processing financial data of various factors accumulated over a period of time, you can be sufficiently prepared for most data types that you will see in real-world work.
- Financial data analysis is the most effective practice in data analysis.

Financial Data Analysis Mini Project

Now, let's use all the skills we have learned so far to solve the following tasks.

- Obtaining financial data from remote data repositories
- Visualizing stock price data based on time series data
- Visualizing trading volume data based on time series data
- Measuring simple daily rate
- Calculating simple daily cumulative rate of return
- Calculating the rate of return by moving the period by month
- Calculating moving average
- Analyzing the correlation between each financial data factor Calculating stock price volatility

Financial Data Analysis Mini Project

- The data material used for data analysis is stock data that track various sectors and economic conditions. It is not simply analyzing the stock price of a specific company. It was chosen because there is no better target data to understand each economic situation as a real economy. In addition, the target data is not for companies, raw materials, bonds, or in a specific country, but for the price of agricultural products around the world.
- Since the US market has a very large impact on the real economy of the world, it was selected as a good practice with meaningful data.

Ticker name

- SPY, which tracks the leading US index S&P 500
- IYW, which tracks U.S. technology company market capitalization index
- VT, which invests in companies around the world
- DBA, which tracks the supply and demand of agricultural products and prices
- US Bond Rate TLT
- PDBC, which tracks the supply and demand of other raw materials and prices
- Gold IAU

Let's code

Step 1

Let's prepare for data acquisition and create data acquisition functions.

```python
import pandas as pd
import numpy as np
import datetime
import matplotlib.pyplot as plt
import pandas_datareader.data as web

from datetime import date, datetime, time, timezone
```

Step 1

A function that obtains the stock price for a given stock when the ticker code is input

```
1 def get_stock_data(ticker,start,end):
2     data = web.DataReader(ticker,'yahoo',start,end)
3     data.insert(0,"Ticker", ticker)
4     return data
```

Line 2, 3

- 2: Read the stock data from Yahoo Finance.
- 3: Create a column name called Ticker at index 0 of the data frame called data created above, and put the ticker name for the data element.

Step 1

- To test whether data acquisition works properly, enter Disney's ticker code to test the function.
- A ticker search can be easily done through Yahoo Finance or investing.com

```
1 ticker = 'DIS'
2 start = datetime(2020,1,1)
3 end = datetime.today()
```

Line 1~3

- 1: Tiker code is obtained through search. Search for the ticker code of the company you are interested in and use it.
- 2: Specify the start date as a timestamp.
- 3: Specify to acquire data up to today. It can be specified as a specific date.

Step 1

```
1 d = get_stock_data(ticker,start,end)
2 d.head()
```

Date	Ticker	High	Low	Open	Close	Volume	Adj Close
2019-12-31	DIS	144.770004	143.259995	143.669998	144.630005	5662900	144.630005
2020-01-02	DIS	148.199997	145.100006	145.289993	148.199997	9502100	148.199997
2020-01-03	DIS	147.899994	146.050003	146.399994	146.500000	7320200	146.500000
2020-01-06	DIS	146.029999	144.309998	145.539993	145.649994	8262500	145.649994
2020-01-07	DIS	146.869995	145.419998	145.990005	145.699997	6906500	145.699997

Line 1~2

- 1: Store the high price, low price, open price, close price, trading volume, and modified closing price data for the ticker in the data frame called d.
- 2: Check the data to see if the function works properly.

Step 1

Since the analysis will be based on only the closing price for each ticker, pivot in the required form.

```
d=d.pivot(index=None, columns="Ticker", values="Close")
d.head()
```

Ticker	DIS
Date	
2019-12-31	144.630005
2020-01-02	148.199997
2020-01-03	146.500000
2020-01-06	145.649994
2020-01-07	145.699997

Step 2

- Create a data frame required for analysis using the created function.
- The name of the data frame is the name of each ticker: SPY / IYW / VT / DBA / TLT / PDBC / IAU

```
SPY = get_stock_data("SPY",start,end)
IYW = get_stock_data("IYW",start,end)
VT = get_stock_data("VT",start,end)
DBA = get_stock_data("DBA",start,end)
TLT = get_stock_data("TLT",start,end)
PDBC = get_stock_data("PDBC",start,end)
IAU = get_stock_data("IAU",start,end)
```

Step 2

```
1 SPY.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 470 entries, 2019-12-31 to 2021-11-09
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Ticker     470 non-null    object
 1   High       470 non-null    float64
 2   Low        470 non-null    float64
 3   Open       470 non-null    float64
 4   Close      470 non-null    float64
 5   Volume     470 non-null    float64
 6   Adj Close  470 non-null    float64
dtypes: float64(6), object(1)
memory usage: 29.4+ KB
```

Line 1

- It is a DatetimeIndex and there is no NaN.

Step 2

```
1 # Execute pivoting per data frame
2 SPY=SPY.pivot(index=None, columns="Ticker", values="Close")
3 IYW=IYW.pivot(index=None, columns="Ticker", values="Close")
4 VT=VT.pivot(index=None, columns="Ticker", values="Close")
5 DBA=DBA.pivot(index=None, columns="Ticker", values="Close")
6 TLT=TLT.pivot(index=None, columns="Ticker", values="Close")
7 PDBC=PDBC.pivot(index=None, columns="Ticker", values="Close")
8 IAU=IAU.pivot(index=None, columns="Ticker", values="Close")
```

Line 1

- Execute pivoting per data frame.

Step 2

- Each created data frame is combined into one data frame for efficient analysis.
- At this time, if the configuration and properties of the data frames to be merged are the same, data consistency can be maintained regardless of the direction of the row or column. Remember that this is something you must check before merging data frames.
- Each data frame we're going to merge now has the same index, the same column, and the same type of data elements. Therefore, we use a function concat() to concatenate while maintaining the shape of the existing data frame.

```
pandas.concat(objs, axis=0, join='outer', ignore_index=False, keys=None, levels=None, names=None,
verify_integrity=False, sort=False, copy=True)
```

- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.concat.html

Step 2

```
1 stock=pd.concat([SPY,IYW,VT,DBA,TLT,PDBC,IAU],
2                 axis=1,
3                 join='outer')
4 stock.head()
```

Ticker	SPY	IYW	VT	DBA	TLT	PDBC	IAU
Date							
2019-12-31	321.859985	58.150002	80.989998	16.559999	135.479996	16.559999	29.000000
2020-01-02	324.869995	59.355000	81.809998	16.500000	137.009995	16.639999	29.219999
2020-01-03	322.410004	58.762501	81.070000	16.309999	139.119995	16.780001	29.620001
2020-01-06	323.640015	59.125000	81.370003	16.350000	138.330002	16.799999	29.920000
2020-01-07	322.730011	59.147499	81.120003	16.389999	137.649994	16.770000	30.040001

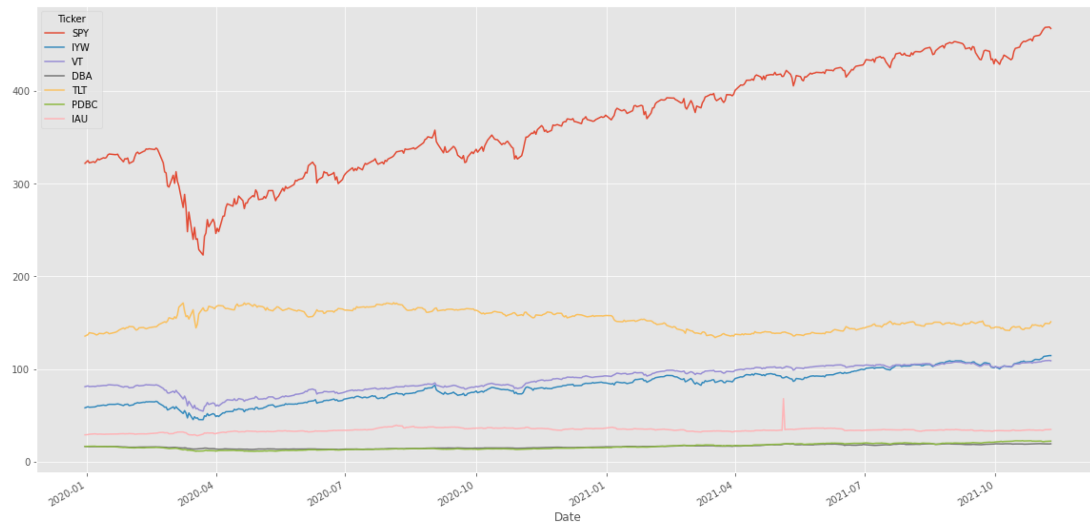
Line 1~3

- 1: List the data frames to be concatenated.
- 2: The default option is axis=0. Each data frame is concatenated vertically.
- 3: If the column name is inner, the standard is the intersection of each data frame. The default value is outer.

Step 3

- If you draw a graph based on the closing price data on a time series basis (date), you can check the stock price movement. By overlapping the graphs of each ticker, you can see the approximate correlation.

```
1 plt.style.use('ggplot')
2 stock.plot(figsize = (20,10))
3 plt.show()
```



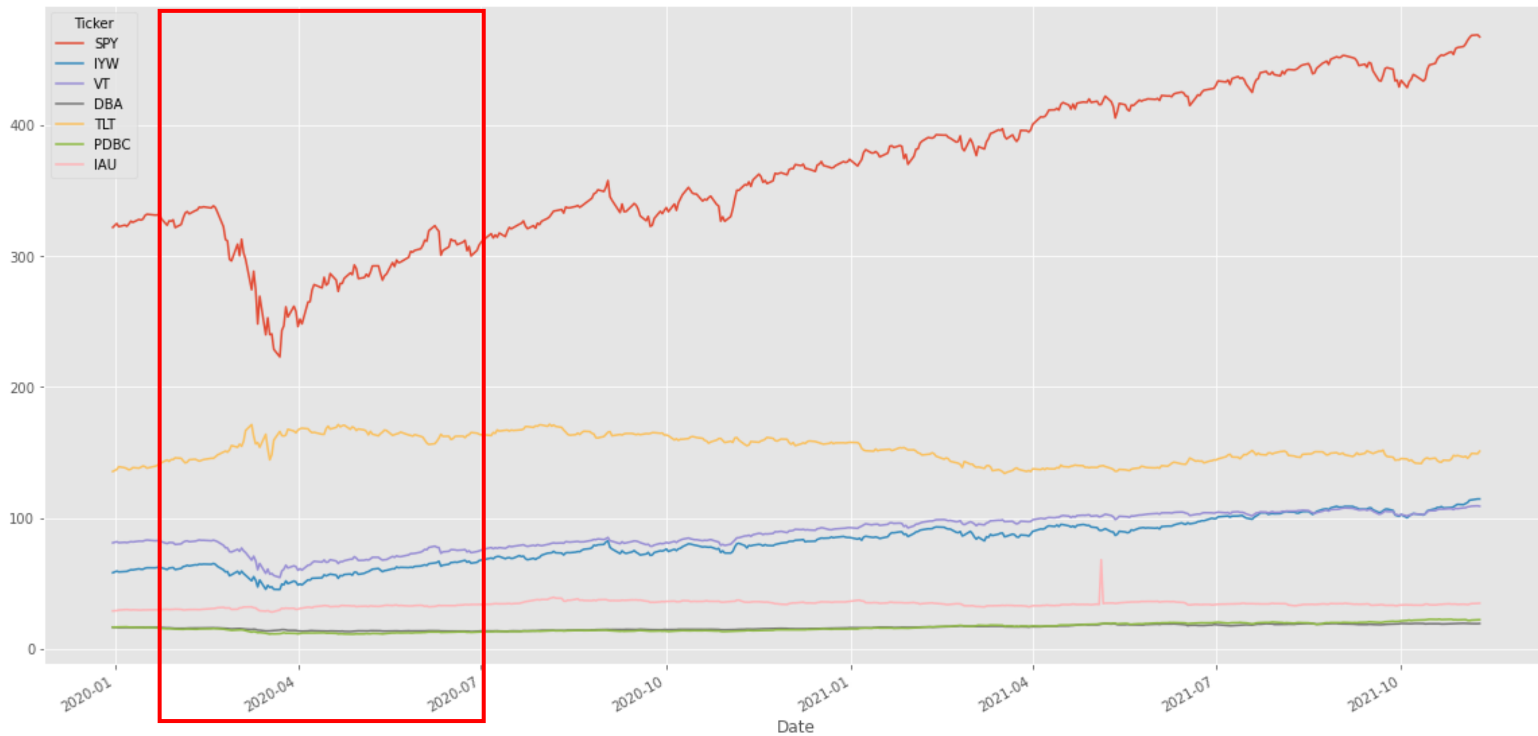
Line 1

- Specify the graph style.

Step 3

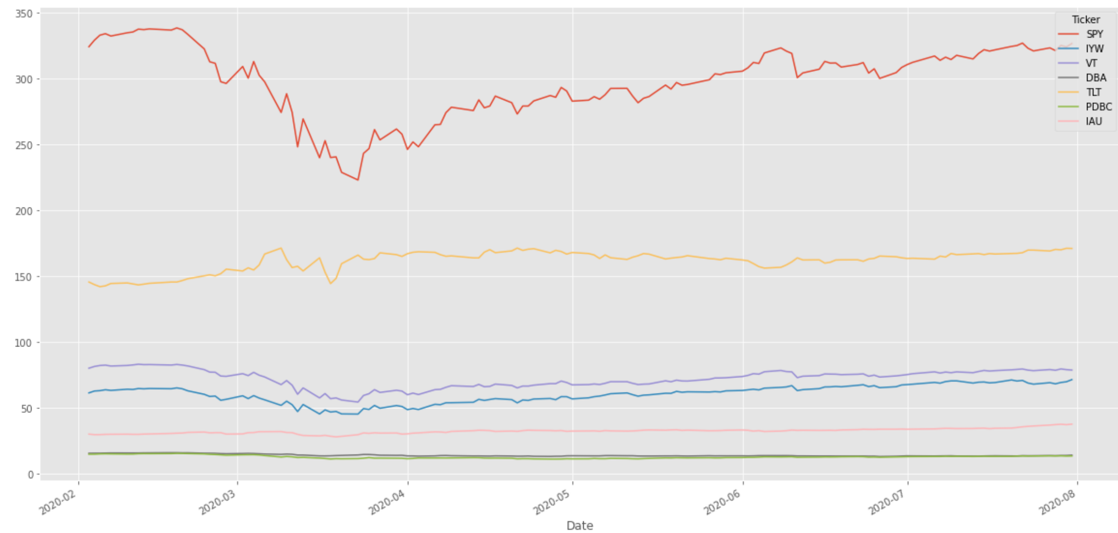
- If you see the graph below, there is a section with serious price fluctuations. This is the period when Corona started. Let's check the data by slicing using the time series data index only for this period.

```
covid=stock['2020-2-1':'2020-7-31']
```



Step 3

```
1 plt.style.use('ggplot')
2 covid.plot(figsize = (20,10))
3 plt.show()
```



Line 1~3

- 1: Specify the graph style.
- 3: There are factors that have a large impact on price fluctuations in the aftermath of the corona shock, while there are factors that rise to that point or have little effect.

Step 3

Let's separate the relevant period factors to check the graph.

```
1 x = covid.index
2 s_y = covid[['SPY']]
3 i_y = covid[['IAU']]
4 d_y = covid[['DBA']]
5 t_y = covid[['TLT']]
```

Line 2

- Select three pieces of data with different personalities and compare them.

Step 3

- Let's compare how assets with different personalities react during times of great shock to economic conditions like the coronavirus.
- You can see that the general stock price declines sharply and then gradually recovers, but gold and bonds are the opposite. In particular, in the case of bonds, you can see that the trend is opposite to the stock price.

```
1 import matplotlib.pyplot as plt
2
3 fig, axs = plt.subplots(1, 3, figsize=(15, 5))
4 axs[0].plot(x, s_y)
5 axs[1].plot(x, i_y)
6 axs[2].plot(x, t_y)
7
8
9 fig.suptitle('Covid 19')
```

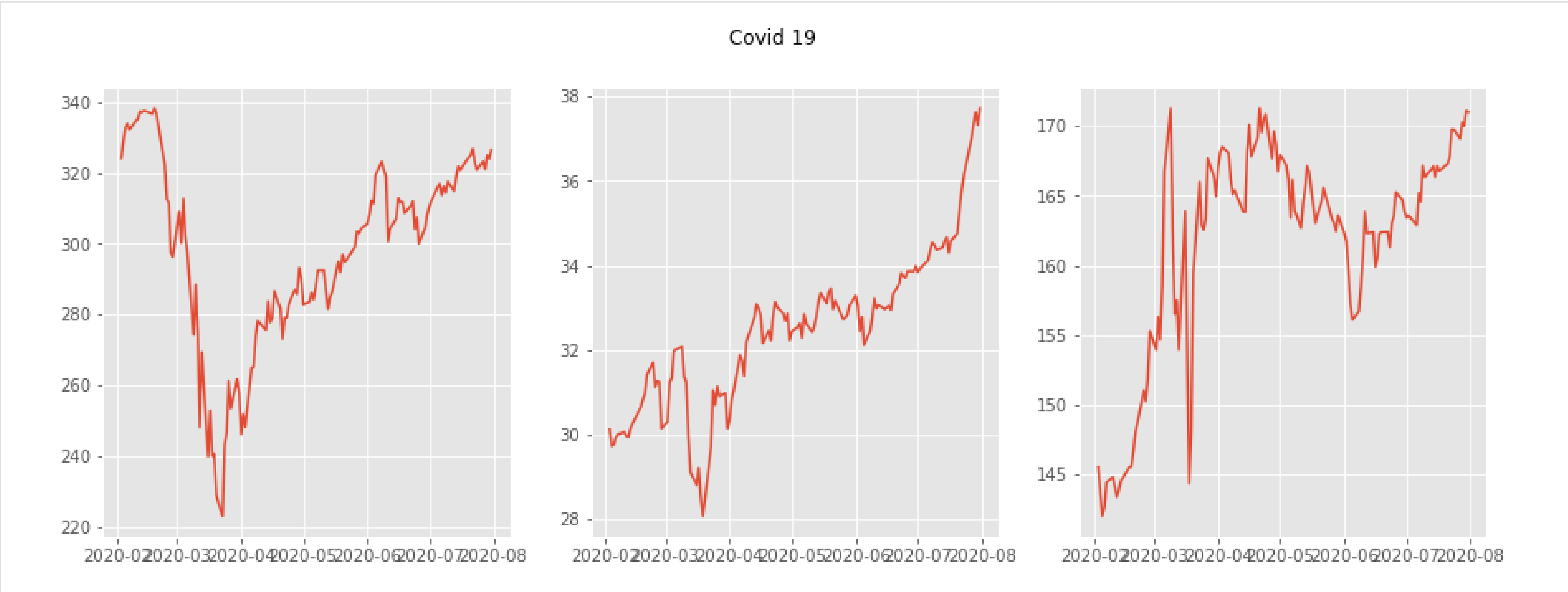
```
Text(0.5, 0.98, 'Covid 19')
```

Line 4~6

- 4: SPY
- 5: IAU
- 6: TLT

Step 3

These graphs are the result of the previous code.



Step 4

Let's visualize the trading volume data in the form of a bar graph for a specific ticker.

```
import pandas as pd
import numpy as np
import datetime
import matplotlib.pyplot as plt
import pandas_datareader.data as web

from datetime import date, datetime, time, timezone
```

```
def get_stock_data(ticker,start,end):
    data = web.DataReader(ticker,'yahoo',start,end)
    data.insert(0,"Ticker", ticker)
    return data
```

Step 4

```
1 ticker = 'PDBC'
2 start = datetime(2020,1,1)
3 end = datetime.today()
```

Line 1~3

- Statistics on raw materials
- Specify the start date with a timestamp.
- Specify to acquire data up to today.

Step 4

```
1 df = get_stock_data(ticker,start,end)
2 df.head()
```

Date	Ticker	High	Low	Open	Close	Volume	Adj Close
2019-12-31	PDBC	16.650000	16.510000	16.520000	16.559999	1780800.0	16.558916
2020-01-02	PDBC	16.670000	16.520000	16.570000	16.639999	4004600.0	16.638912
2020-01-03	PDBC	16.840000	16.709999	16.809999	16.780001	760200.0	16.778904
2020-01-06	PDBC	16.910000	16.770000	16.900000	16.799999	1608700.0	16.798901
2020-01-07	PDBC	16.799999	16.716999	16.750000	16.770000	1723200.0	16.768904

Line 1

- In a data frame called df, the high price, low price, open price, closing price, trading volume, and modified closing price data for the corresponding ticker are stored.

Step 4

```
1 df.drop(['Ticker','High','Low','Open',"Close","Adj Close"], axis=1, inplace=True)
```

Line 1

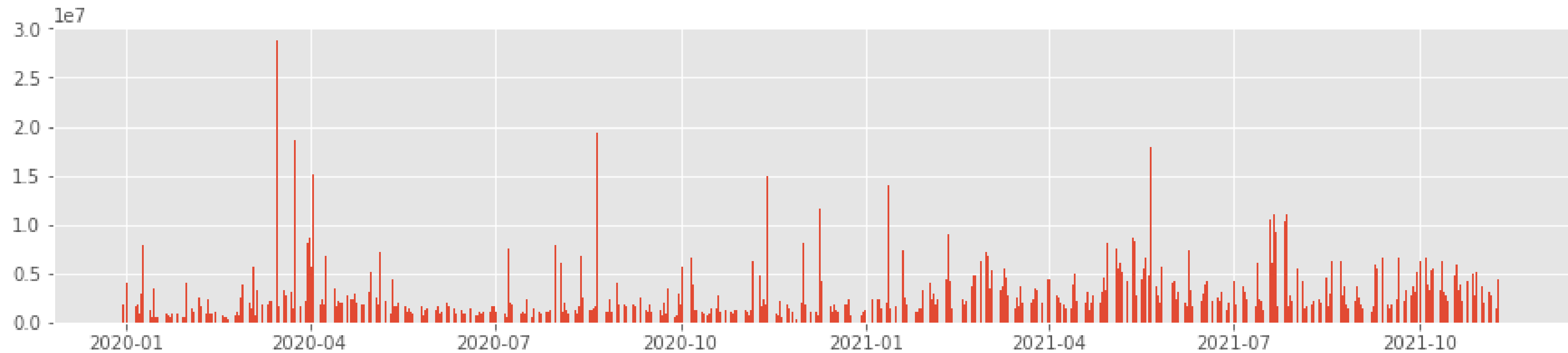
- Delete all except the volume column.

```
df.head()
```

	Volume
Date	
2019-12-31	1780800.0
2020-01-02	4004600.0
2020-01-03	760200.0
2020-01-06	1608700.0
2020-01-07	1723200.0

Step 4

```python
x = df.index
y = df['Volume']
plt.figure(figsize = (15,3))
plt.bar(x,y)
plt.show()
```



Step 5

- Using matplotlib's subplot2grid, the graph of the closing price is visualized in the upper layout and the trading volume in the same period is visualized in the lower layout.

```
ax = subplot2grid((nrows, ncols), (row, col), rowspan, colspan)
```

- https://matplotlib.org/stable/gallery/userdemo/demo_gridspec01.html#sphx-glr-gallery-userdemo-demo-gridspec01-py

```
1 ticker = 'PDBC'
2 start = datetime(2020,1,1)
3 end = datetime.today()
```

Line 1~3

- Statistics on raw materials
- Specify the start date with a timestamp.
- Specify to acquire data up to today.

```
df = get_stock_data(ticker,start,end)
```

Step 5

```
ax = subplot2grid((nrows, ncols), (row, col), rowspan, colspan)
```

```
1  fig = plt.figure(figsize=(12, 8))
2
3  top_grid = plt.subplot2grid((4,4), (0,0), rowspan=3, colspan=4)
4  bottom_grid = plt.subplot2grid((4,4), (3,0), rowspan=1, colspan=4)
5
6  top_grid.plot(df.index, df['Close'], label='Close')
7  bottom_grid.plot(df.index, df['Volume'], label='Volume')
8
9  plt.tight_layout()
10
11 plt.legend()
12 plt.show()
```

Line 1~3

- Set the range, start position, and occupied range of the upper area grid.
- Set the range, start position, and occupied range of the lower area grid.
- Show closing price data on the upper grid.

Step 5

ax = subplot2grid((nrows, ncols), (row, col), rowspan, colspan)

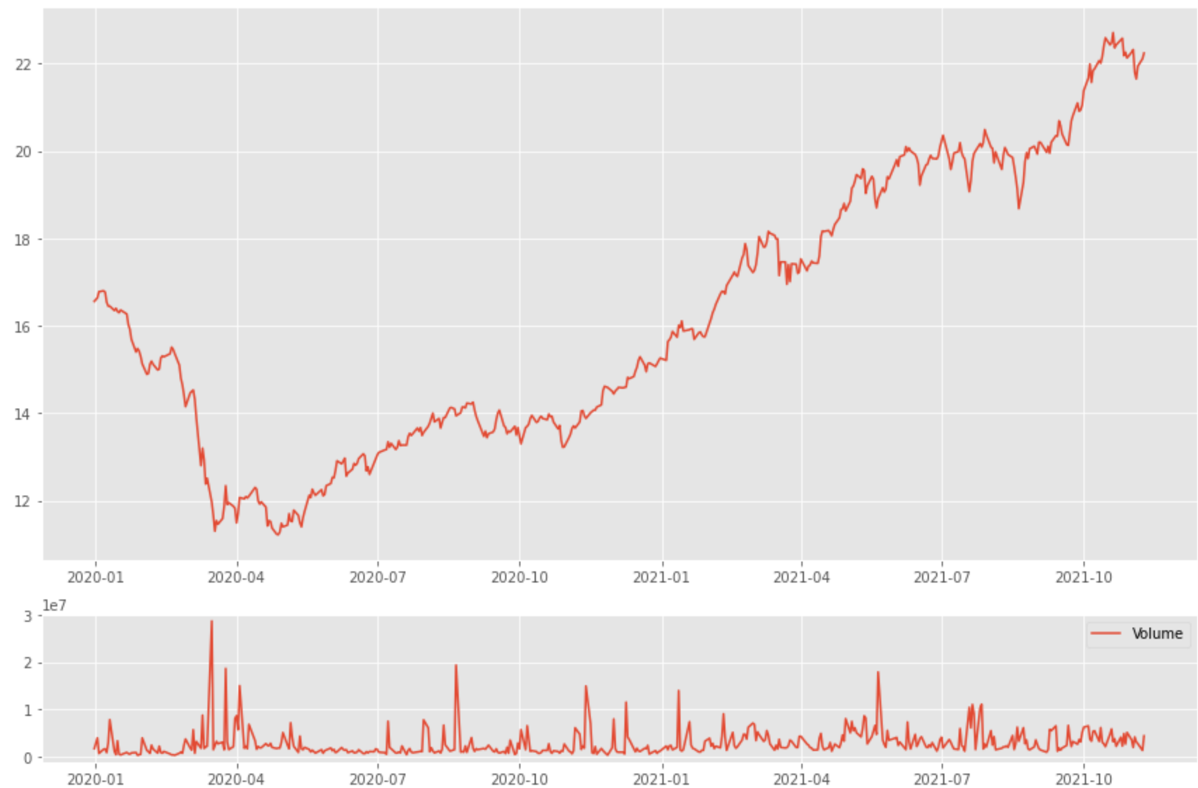
```
1  fig = plt.figure(figsize=(12, 8))
2
3  top_grid = plt.subplot2grid((4,4), (0,0), rowspan=3, colspan=4)
4  bottom_grid = plt.subplot2grid((4,4), (3,0), rowspan=1, colspan=4)
5
6  top_grid.plot(d                                        lose')
7  bottom_grid.plot(df.index, df['Volume'], label='Volume')
8
9  plt.tight_layout()
10
11 plt.legend()
12 plt.show()
```

Line 1~3

- Show volume data on the lower grid.
- A function that allows subplots to be printed at the maximum size in the figure

Step 5

```
ax = subplot2grid((nrows, ncols), (row, col), rowspan, colspan)
```



Step 6

- **We will use the pandas.Series.shift() method to calculate the daily percentage change.**
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.shift.html?highlight=shift#pandas.Series.shift
- Comparing the current data with the data at a specific point in time to calculate the percentage change is also a common task in daily work.
- If you solve this with Excel, you can do it as shown below. If you see the image, you can understand why the shift() method is useful.

	A	B	C
4			
5	**Year**	**Total Asset Size (in millions)**	
6	2018	$375	
7	2017	$365	
8			
9	Percentage Change is calculated using the formula given below		
10	**Percentage Change = (New Value - Original Value) / Original Value * 100**		
11			
12	**Percentage Change Formula**	=(B6-B7)/B7*100	
13	**Percentage Change**	**2.74**	
14			

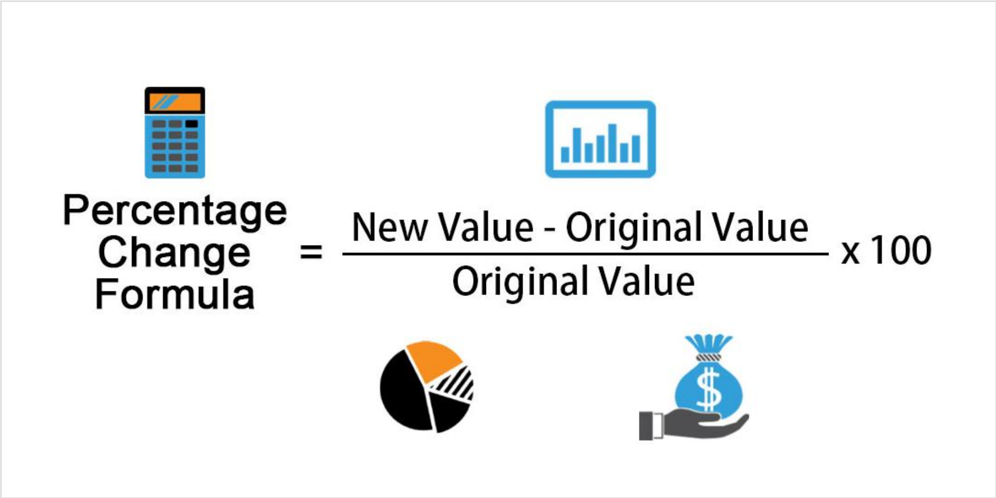
Step 6

```
pandas.Series.shift()
```

- The formula for calculating the daily percentage change based on the closing price is simple

Today's daily percentage change = ((Today's closing price - Yesterday's closing price) / Yesterday's closing price) * 100

Daily Percentage Change = (New Value – Original Value) / Original Value * 100



https://www.educba.com/percentage-change-formula/

Step 6

pandas.Series.shift()

```
stock.head()
```

Ticker	SPY	IYW	VT	DBA	TLT	PDBC	IAU
Date							
2019-12-31	321.859985	58.150002	80.989998	16.559999	135.479996	16.559999	29.000000
2020-01-02	324.869995	59.355000	81.809998	16.500000	137.009995	16.639999	29.219999
2020-01-03	322.410004	58.762501	81.070000	16.309999	139.119995	16.780001	29.620001
2020-01-06	323.640015	59.125000	81.370003	16.350000	138.330002	16.799999	29.920000
2020-01-07	322.730011	59.147499	81.120003	16.389999	137.649994	16.770000	30.040001

Step 6

pandas.Series.shift()

```
stock['SPY']
```

```
Date
2019-12-31    321.859985
2020-01-02    324.869995
2020-01-03    322.410004
2020-01-06    323.640015
2020-01-07    322.730011
                 ...
2021-11-03    464.720001
2021-11-04    466.910004
2021-11-05    468.529999
2021-11-08    468.929993
2021-11-09    467.380005
Name: SPY, Length: 470, dtype: float64
```

Step 6

pandas.Series.shift()

```
1  stock['SPY'].shift(1)
```

```
Date
2019-12-31           NaN
2020-01-02    321.859985
2020-01-03    324.869995
2020-01-06    322.410004
2020-01-07    323.640015
                 ...
2021-11-03    461.899994
2021-11-04    464.720001
2021-11-05    466.910004
2021-11-08    468.529999
2021-11-09    468.929993
Name: SPY, Length: 470, dtype: float64
```

Line 1

- Shift one day to get the previous day's closing price.

Step 6

pandas.Series.shift()

```
spy_dayily_pc=(stock['SPY']/stock['SPY'].shift(1)-1)*100
```

```
spy_dayily_pc
```

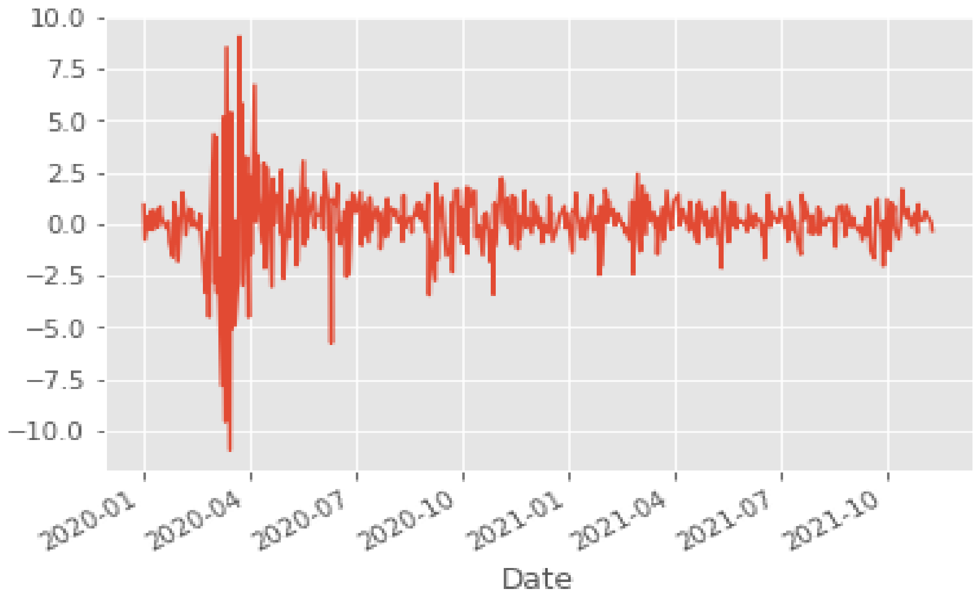
```
Date
2019-12-31          NaN
2020-01-02     0.935192
2020-01-03    -0.757223
2020-01-06     0.381505
2020-01-07    -0.281178
                 ...
2021-11-03     0.610523
2021-11-04     0.471252
2021-11-05     0.346961
2021-11-08     0.085372
2021-11-09    -0.330537
Name: SPY, Length: 470, dtype: float64
```

Step 6

pandas.Series.shift()

```
spy_dayily_pc.plot()
```

<AxesSubplot:xlabel='Date'>



Step 6

- A histogram is a graph showing the frequency distribution.
- It represents the frequency of data values by time.
- At this time, the number of sections is used by setting the parameter bins value of the hist() function.
- For reference, the default value of hist() is 10. The shape of the graph changes depending on the bins. You should set the bins value by carefully checking the characteristics of the data.

Step 6

pandas.DataFrame.hist( )

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.hist.html?highlight=hist#pandas.DataFrame.hist

```
1 spy_dayily_pc=(stock['SPY']-stock['SPY'].shift(1))/stock['SPY'].shift(1) * 100
```

Line 1

- Don't confuse the new formula. It's just written out to help you understand.

```
1 spy_dayily_pc.iloc[0] =0
```

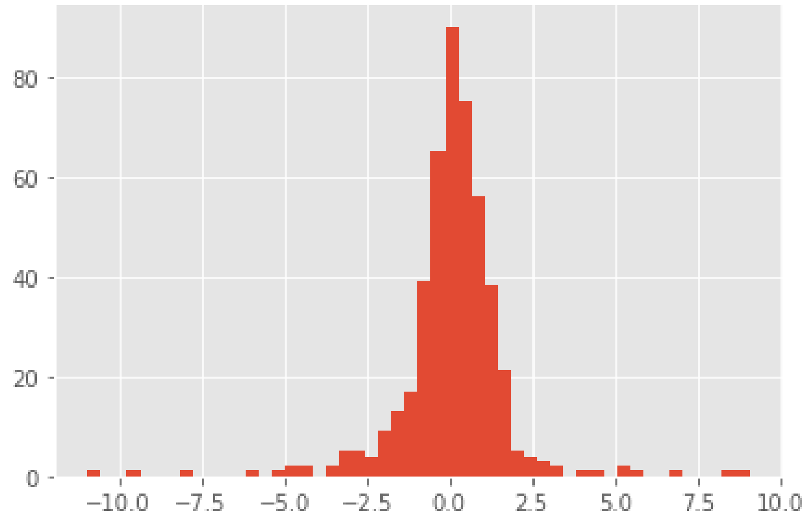
Line 1

- Since the first value is missing data, it is replaced with 0.

Step 6

pandas.DataFrame.hist( )

```
1 plt.hist(spy_dayily_pc, bins = 50)
2 plt.show()
```



Line 1

- The frequency is expressed by dividing the daily percentage change of the stock price into 50 sections.

Step 7

Let's create a new data frame that calculates daily stock price changes for all tickers, calculate daily cumulative returns, and analyze the correlation.

```
stock_dayily_pc=(stock-stock.shift(1))/stock.shift(1) * 100
```

```
stock_dayily_pc.head()
```

Ticker	SPY	IYW	VT	DBA	TLT	PDBC	IAU
Date							
2019-12-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-02	0.935192	2.072224	1.012470	-0.362316	1.129317	0.483091	0.758618
2020-01-03	-0.757223	-0.998229	-0.904532	-1.151518	1.540034	0.841354	1.368931
2020-01-06	0.381505	0.616889	0.370054	0.245254	-0.567850	0.119181	1.012827
2020-01-07	-0.281178	0.038053	-0.307239	0.244642	-0.491584	-0.178564	0.401072

Step 7

- The formula for the simple daily cumulative return is also simple.
- It can be obtained by accumulating and multiplying the daily stock price change rate obtained above. We can use the .compsum() method.

```
stock_d_cr=stock_dayily_pc.cumsum()
```

Step 7

```
stock_d_cr
```

Ticker	SPY	IYW	VT	DBA	TLT	PDBC	IAU
Date							
2019-12-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-02	0.935192	2.072224	1.012470	-0.362316	1.129317	0.483091	0.758618
2020-01-03	0.177969	1.073995	0.107938	-1.513834	2.669351	1.324445	2.127549
2020-01-06	0.559474	1.690883	0.477992	-1.268580	2.101501	1.443626	3.140376
2020-01-07	0.278296	1.728937	0.170754	-1.023938	1.609917	1.265062	3.541448
...	...	...	...	...	...	...	...
2021-11-03	43.083960	74.822849	35.309758	17.963124	10.244286	31.346321	67.530690
2021-11-04	43.555212	76.397513	35.475703	17.654326	11.295326	30.658249	68.567722
2021-11-05	43.902172	76.626530	35.797836	17.138062	12.790905	31.997745	69.916705
2021-11-08	43.987545	77.224122	36.008849	16.878585	12.603364	32.772586	70.321795
2021-11-09	43.657007	77.224122	35.734196	17.607000	13.905204	33.360552	70.811704

Step 7

```
stock_d_cr.plot(figsize = (20,10))
```

<AxesSubplot:xlabel='Date'>

Step 8

The correlation coefficient refers to measuring the strength of the association between data as we learned earlier. The closer to 1.0, the stronger the relationship, and the closer to 0, the less the relationship. Let's analyze it using the learned .corr().

```
df_corr = stock_dayily_pc.corr()
```

```
df_corr
```

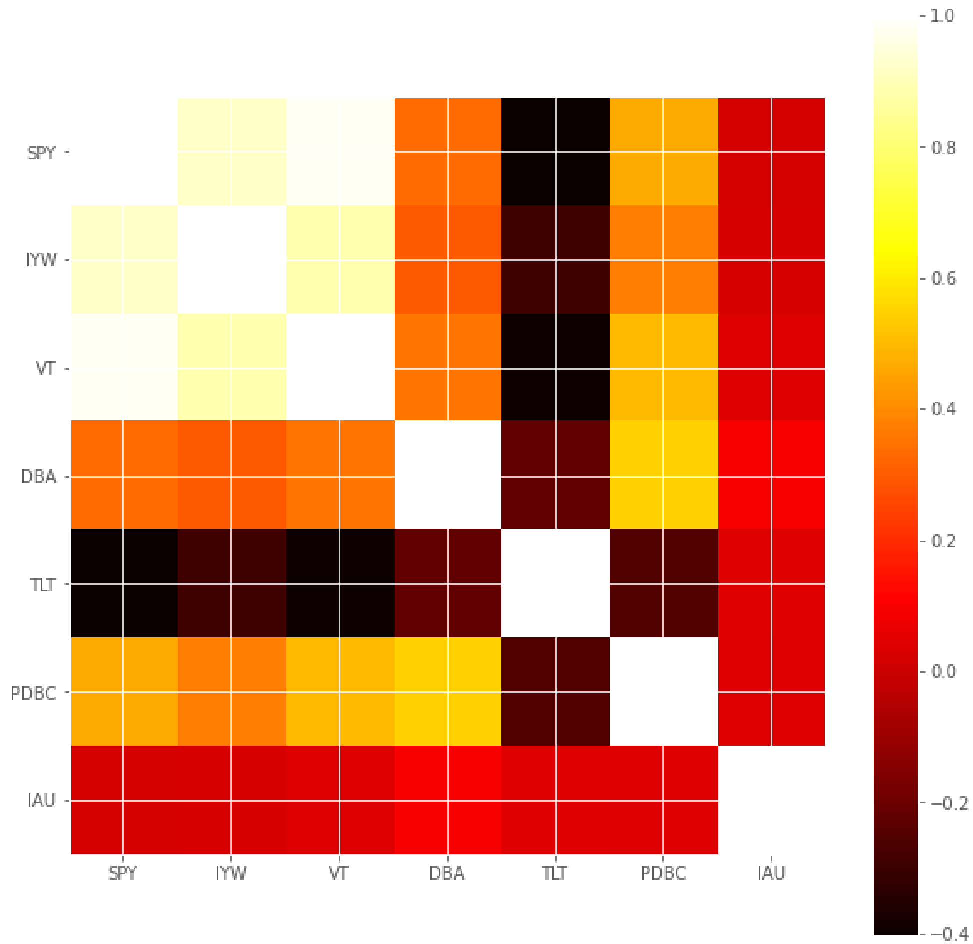
Ticker	SPY	IYW	VT	DBA	TLT	PDBC	IAU
Ticker							
SPY	1.000000	0.918477	0.979306	0.331443	-0.404370	0.464007	0.021511
IYW	0.918477	1.000000	0.889873	0.294046	-0.293439	0.374583	0.027078
VT	0.979306	0.889873	1.000000	0.357680	-0.397998	0.495927	0.038896
DBA	0.331443	0.294046	0.357680	1.000000	-0.218909	0.545092	0.098664
TLT	-0.404370	-0.293439	-0.397998	-0.218909	1.000000	-0.253284	0.043516
PDBC	0.464007	0.374583	0.495927	0.545092	-0.253284	1.000000	0.039487
IAU	0.021511	0.027078	0.038896	0.098664	0.043516	0.039487	1.000000

Step 8

- Let's visualize it as a heatmap.
- The darker the color, the lower the correlation, and the brighter the color, the higher the correlation. Let's check how each economic factor correlates in the real economy.

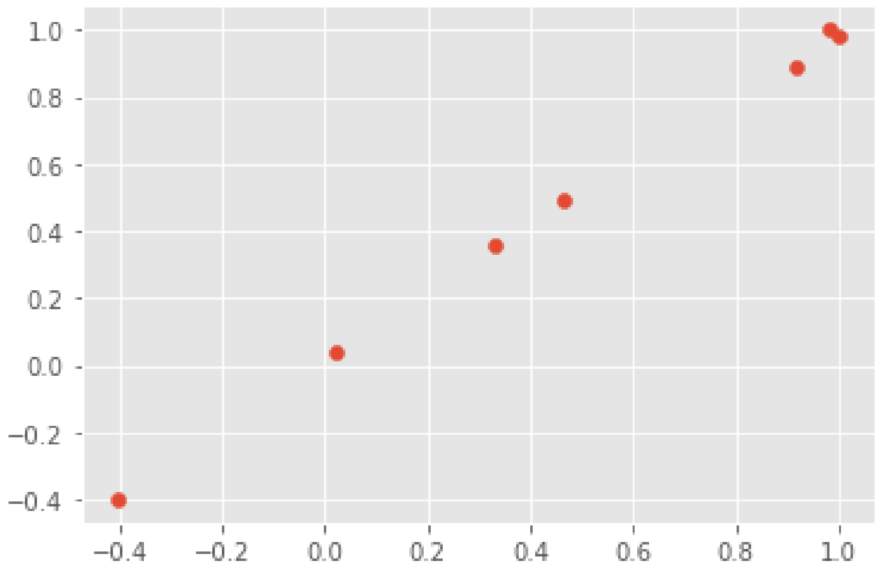
```
plt.imshow(df_corr, cmap ='hot', interpolation ='none')
plt.colorbar()
plt.xticks(range(len(df_corr)),df_corr.columns)
plt.yticks(range(len(df_corr)),df_corr.columns)

plt.gcf().set_size_inches(10,10)
```

Step 8



Step 8

```
plt.scatter(df_corr.SPY,df_corr.VT)
plt.show()
```



Step 8

Ex You can see that US bonds and general stocks are completely uncorrelated by economic conditions.

- When the economy is good, the company's performance improves, and the stock price rises. Conversely, when the economy is bad, the value of bonds rises. It is now possible to be verified through actual data analysis rather than theory.

If you think once more by applying the correlation coefficient in practice, there is a complementary effect for each asset. You can create a strategy to mitigate the risk.

A correlation coefficient close to 1 means that the value rises when it rises and falls when it falls. It means that there is no risk mitigation effect on each other. If the correlation coefficient is close to 0, it means that there is no relationship between the value rises and falls, and on the contrary, the risk mitigation effect is large.

For example, it can be used in making a marketing plan in the real world. As a data-based decision-making tool, it can be used for analyzing the correlation between sales timing and sales by product, setting a product launch time, and building company portfolios.

Step 9

- Let's track the change in stock price over a specific period of time. This technique is usually used to determine the risk rate of the stock by comparing the volatility of the entire market index as reference data.
- The purpose is to measure the amount of change in a specific period and compare it with stable reference data to use it to determine the risk of the currently evaluated data.
- First, the stock price volatility can be obtained by calculating the standard deviation of the stock price volatility through the moving average. The biggest influence on this data is the period of time to be tracked. That is, the size of the window has a large effect.
- If the window is wide, representativeness will be blurred, and if it is too narrow, it will be close to the standard deviation. It is very difficult and important to set the size of the window to be measured.
- Not everything can be concluded through computer calculations. There are many moments when people have to make a decision based on understanding the context and the outcome to achieve.
- It is very important to understand that the results of the data can be different depending on the judgment of the learner, and the decision-making in the real world can be changed accordingly, rather than just learning the skills as a practical task using pandas.

Step 9

```
periods = 75

vol = stock_dayily_pc.rolling(window=periods).std()

vol
```

Ticker Date	SPY	IYW	VT	DBA	TLT	PDBC	IAU
2019-12-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-03	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-06	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-07	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...
2021-11-03	0.684333	0.935314	0.668490	0.786147	0.817607	1.149804	0.816876
2021-11-04	0.680715	0.944153	0.659341	0.768860	0.814283	1.127037	0.824916
2021-11-05	0.681204	0.940708	0.660003	0.764044	0.825371	1.132134	0.839324
2021-11-08	0.672474	0.929907	0.656638	0.762527	0.821952	1.134399	0.839492
2021-11-09	0.673932	0.930004	0.657473	0.718107	0.834575	1.132586	0.840922

Step 9

```python
vol["SPY"].plot()
vol["TLT"].plot()
vol["DBA"].plot()
```

```
<AxesSubplot:xlabel='Date'>
```