004 - Import Data for Multiple Stocks

December 8, 2023

1 #004 - Import Data for Multiple Assets

On this code we import data from YFinance, export and read a CSV file, then perform some analysis and organize the DataFrame.

From #001;

Import libraries

- Link to Pandas Documentation: https://pandas.pydata.org/docs/index.html
- Link to datareader Documentation: https://pandas-datareader.readthedocs.io/en/latest/index.html
- Link to Yf Documentation: https://pandas-datareader.readthedocs.io/en/latest/readers/yahoo.html
- Link to Plotly Documentation: https://plotly.com/python/

[1]:
```
#    !pip install pandas
#    !pip install pandas-datareader
#    !pip install numpy
#    !pip install yfinance
#    !pip install datetime
#    !pip install plotly_express
```

[2]:
```
#import Libraries
import pandas as pd
from pandas_datareader import data as pdr
import numpy as np
import yfinance as yf
import datetime as dt
import plotly.express as px
import plotly.graph_objects as go
```

1.1 4.1 Define a portfolio, obtain and export data

[3]:
```
# Code to obtain stock data using Pandas Datareader
# Select the start/end dates and ticker symbols
# Data is saved in a stock_data.csv file

tickers = [item for item in input("Enter the stock tickers, for portfolio␣
 ↪(space them only) : ").split()]
```

```
file_name = "Portfolio.csv"

yf.pdr_override()
```

```
Enter the stock tickers, for portfolio (space them only) : AAPL AMZN GOOG META
MSFT NVDA TSLA
```

[4]:
```
# Define the start and end dates, last 5 years
end = dt.datetime.now()
start = end - dt.timedelta(days = 365*5)
```

[5]:
```
#obtain data from Yahoo Finance
df = pdr.get_data_yahoo(tickers, start = start, end = end)
df
```

```
[**********100%%**********]  7 of 7 completed
```

[5]:

	Adj Close					\
	AAPL	AMZN	GOOG	META	MSFT	
Date						
2018-12-10	40.695984	82.051498	51.977501	141.850006	102.167152	
2018-12-11	40.463226	82.162003	52.587502	142.080002	103.116745	
2018-12-12	40.576008	83.177002	53.183998	144.500000	103.582054	
2018-12-13	41.019913	82.918999	53.095001	145.009995	103.933426	
2018-12-14	39.707371	79.595497	52.105000	144.059998	100.685776	
…	…	…	…	…	…	
2023-12-04	189.429993	144.839996	130.630005	320.019989	369.140015	
2023-12-05	193.419998	146.880005	132.389999	318.290009	372.519989	
2023-12-06	192.320007	144.520004	131.429993	317.450012	368.799988	
2023-12-07	194.270004	146.880005	138.449997	326.589996	370.950012	
2023-12-08	195.340103	147.270004	136.460999	330.920013	373.250000	

			Close			…	\
	NVDA	TSLA	AAPL	AMZN	GOOG	…	
Date						…	
2018-12-10	37.670109	24.343332	42.400002	82.051498	51.977501	…	
2018-12-11	36.759731	24.450666	42.157501	82.162003	52.587502	…	
2018-12-12	36.935837	24.440001	42.275002	83.177002	53.183998	…	
2018-12-13	36.933376	25.119333	42.737499	82.918999	53.095001	…	
2018-12-14	36.328114	24.380667	41.369999	79.595497	52.105000	…	
…	…	…	…	…	…	…	
2023-12-04	455.059998	235.580002	189.429993	144.839996	130.630005	…	
2023-12-05	465.660004	238.720001	193.419998	146.880005	132.389999	…	
2023-12-06	455.029999	239.369995	192.320007	144.520004	131.429993	…	
2023-12-07	465.959991	242.639999	194.270004	146.880005	138.449997	…	
2023-12-08	476.250092	242.354996	195.340103	147.270004	136.460999	…	

```
                 Open                                  Volume              \
                 MSFT        NVDA        TSLA        AAPL        AMZN
Date
2018-12-10  104.800003   36.450001   24.000000   248104000   149896000
2018-12-11  109.800003   38.889999   24.660667   189126800   124894000
2018-12-12  110.889999   37.105000   24.628000   142510800   131960000
2018-12-13  109.580002   37.697498   24.676666   127594400   105426000
2018-12-14  108.250000   36.802502   25.000000   162814800   127344000
…                  …           …           …           …           …
2023-12-04  369.100006  460.769989  235.750000    43389500    48294200
2023-12-05  366.450012  454.660004  233.869995    66628400    46822400
2023-12-06  373.540009  472.149994  242.919998    41089700    39679000
2023-12-07  368.230011  457.000000  241.550003    47433900    52320500
2023-12-08  369.200012  465.950012  240.270004    25404775    20995189


                GOOG      META      MSFT      NVDA       TSLA
Date
2018-12-10  36154000  26422200  40801500  62947200   99202500
2018-12-11  27894000  20300300  42381900  67191200   94632000
2018-12-12  30476000  23696900  36183000  65413600   75405000
2018-12-13  26596000  18148600  31333400  47138400  110488500
2018-12-14  33732000  21785800  47043100  47182000   95064000
…                 …         …         …         …          …
2023-12-04  24117100  19037100  32063300  43754300  104099800
2023-12-05  19235100  16952100  23065000  37171800  137971100
2023-12-06  16360600  11294300  21182100  38059000  126436200
2023-12-07  38297500  15891500  23099800  35000100  106949500
2023-12-08  14086832   6687972   9610302  24494477   70541618

[1259 rows x 42 columns]
```

1.2 4.2 Manipulate and Organize Data

[6]:
```
#select Ajusted Close Prices
Adj_Close_df = df['Adj Close']
Adj_Close_df
```

[6]:
```
                 AAPL        AMZN        GOOG        META        MSFT  \
Date
2018-12-10  40.695984   82.051498   51.977501  141.850006  102.167152
2018-12-11  40.463226   82.162003   52.587502  142.080002  103.116745
2018-12-12  40.576008   83.177002   53.183998  144.500000  103.582054
2018-12-13  41.019913   82.918999   53.095001  145.009995  103.933426
2018-12-14  39.707371   79.595497   52.105000  144.059998  100.685776
…                 …           …           …           …           …
2023-12-04  189.429993  144.839996  130.630005  320.019989  369.140015
```

```
2023-12-05  193.419998  146.880005  132.389999  318.290009  372.519989
2023-12-06  192.320007  144.520004  131.429993  317.450012  368.799988
2023-12-07  194.270004  146.880005  138.449997  326.589996  370.950012
2023-12-08  195.340103  147.270004  136.460999  330.920013  373.250000

                 NVDA        TSLA
Date
2018-12-10   37.670109   24.343332
2018-12-11   36.759731   24.450666
2018-12-12   36.935837   24.440001
2018-12-13   36.933376   25.119333
2018-12-14   36.328114   24.380667
…                …           …
2023-12-04  455.059998  235.580002
2023-12-05  465.660004  238.720001
2023-12-06  455.029999  239.369995
2023-12-07  465.959991  242.639999
2023-12-08  476.250092  242.354996

[1259 rows x 7 columns]
```

[7]:
```
volume_df = df['Volume']
volume_df
```

[7]:
```
                AAPL       AMZN      GOOG      META      MSFT      NVDA  \
Date
2018-12-10  248104000  149896000  36154000  26422200  40801500  62947200
2018-12-11  189126800  124894000  27894000  20300300  42381900  67191200
2018-12-12  142510800  131960000  30476000  23696900  36183000  65413600
2018-12-13  127594400  105426000  26596000  18148600  31333400  47138400
2018-12-14  162814800  127344000  33732000  21785800  47043100  47182000
…                 …          …         …         …         …         …
2023-12-04   43389500   48294200  24117100  19037100  32063300  43754300
2023-12-05   66628400   46822400  19235100  16952100  23065000  37171800
2023-12-06   41089700   39679000  16360600  11294300  21182100  38059000
2023-12-07   47433900   52320500  38297500  15891500  23099800  35000100
2023-12-08   25404775   20995189  14086832   6687972   9610302  24494477

                TSLA
Date
2018-12-10   99202500
2018-12-11   94632000
2018-12-12   75405000
2018-12-13  110488500
2018-12-14   95064000
…                …
2023-12-04  104099800
```

```
2023-12-05  137971100
2023-12-06  126436200
2023-12-07  106949500
2023-12-08   70541618

[1259 rows x 7 columns]
```

[8]:
```
#calculate percentage daily return
p_change_df = Adj_Close_df.pct_change() * 100
p_change_df.replace(np.nan, 0, inplace = True)
p_change_df
```

[8]:
```
                AAPL      AMZN      GOOG      META      MSFT      NVDA  \
Date
2018-12-10  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2018-12-11 -0.571942  0.134677  1.173586  0.162140  0.929450 -2.416711
2018-12-12  0.278726  1.235363  1.134293  1.703264  0.451245  0.479072
2018-12-13  1.094008 -0.310186 -0.167338  0.352937  0.339221 -0.006661
2018-12-14 -3.199768 -4.008130 -1.864585 -0.655125 -3.124741 -1.638796
…                …         …         …         …         …         …
2023-12-04 -0.946461 -1.489494 -2.017703 -1.477747 -1.433872 -2.683629
2023-12-05  2.106322  1.408457  1.347313 -0.540585  0.915635  2.329365
2023-12-06 -0.568706 -1.606754 -0.725135 -0.263909 -0.998604 -2.282782
2023-12-07  1.013933  1.632992  5.341250  2.879188  0.582978  2.402038
2023-12-08  0.550831  0.265522 -1.436619  1.325827  0.620026  2.208366

                TSLA
Date
2018-12-10  0.000000
2018-12-11  0.440918
2018-12-12 -0.043622
2018-12-13  2.779594
2018-12-14 -2.940630
…                …
2023-12-04 -1.360801
2023-12-05  1.332880
2023-12-06  0.272283
2023-12-07  1.366088
2023-12-08 -0.117459

[1259 rows x 7 columns]
```

[9]:
```
# Function to scale stock prices based on their initial starting price
# The objective of this function is to set all prices to start at a value of 1
def price_scaling(raw_prices_df):
    scaled_prices_df = raw_prices_df.copy()
    for i in raw_prices_df.columns[0:]:
```

```
        scaled_prices_df[i] = raw_prices_df[i]/raw_prices_df[i][0]
    return scaled_prices_df
```

[10]:
```
scaling_df = price_scaling(Adj_Close_df)
scaling_df
```

[10]:

Date	AAPL	AMZN	GOOG	META	MSFT	NVDA	TSLA
2018-12-10	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
2018-12-11	0.994281	1.001347	1.011736	1.001621	1.009294	0.975833	1.004409
2018-12-12	0.997052	1.013717	1.023212	1.018682	1.013849	0.980508	1.003971
2018-12-13	1.007960	1.010573	1.021500	1.022277	1.017288	0.980443	1.031877
2018-12-14	0.975707	0.970068	1.002453	1.015580	0.985500	0.964375	1.001534
…	…	…	…	…	…	…	…
2023-12-04	4.654759	1.765233	2.513203	2.256045	3.613099	12.080135	9.677393
2023-12-05	4.752803	1.790095	2.547064	2.243849	3.646182	12.361525	9.806381
2023-12-06	4.725774	1.761333	2.528594	2.237927	3.609771	12.079339	9.833083
2023-12-07	4.773690	1.790095	2.663652	2.302362	3.630815	12.369489	9.967411
2023-12-08	4.799985	1.794848	2.625386	2.332887	3.653327	12.642652	9.955703

[1259 rows x 7 columns]

[11]:
```
Adj_Close_df.describe().round(2)
```

[11]:

	AAPL	AMZN	GOOG	META	MSFT	NVDA	TSLA
count	1259.00	1259.00	1259.00	1259.00	1259.00	1259.00	1259.00
mean	119.76	126.77	97.70	233.39	228.61	172.09	168.24
std	47.00	31.59	30.24	69.21	73.30	121.18	109.11
min	34.12	67.20	48.81	88.91	89.39	31.52	11.93
25%	72.54	95.05	68.01	179.78	161.48	65.50	45.52
50%	131.62	125.98	99.16	216.88	238.19	142.90	196.63
75%	156.31	158.09	125.50	294.28	286.49	226.40	250.87
max	195.93	186.57	150.71	382.18	382.70	504.05	409.97

[12]:
```
p_change_df.describe().round(2)
```

[12]:

	AAPL	AMZN	GOOG	META	MSFT	NVDA	TSLA
count	1259.00	1259.00	1259.00	1259.00	1259.00	1259.00	1259.00
mean	0.15	0.07	0.10	0.11	0.12	0.25	0.27
std	2.05	2.25	2.01	2.77	1.94	3.27	4.10
min	-12.86	-14.05	-11.10	-26.39	-14.74	-18.45	-21.06
25%	-0.83	-1.10	-0.88	-1.15	-0.85	-1.54	-1.78
50%	0.14	0.10	0.11	0.10	0.12	0.29	0.20
75%	1.25	1.22	1.10	1.43	1.12	1.99	2.22
max	11.98	13.54	10.45	23.28	14.22	24.37	19.89

[13]:
```
scaling_df.describe().round(2)
```

[13]:

	AAPL	AMZN	GOOG	META	MSFT	NVDA	TSLA
count	1259.00	1259.00	1259.00	1259.00	1259.00	1259.00	1259.00
mean	2.94	1.55	1.88	1.65	2.24	4.57	6.91
std	1.15	0.39	0.58	0.49	0.72	3.22	4.48
min	0.84	0.82	0.94	0.63	0.87	0.84	0.49
25%	1.78	1.16	1.31	1.27	1.58	1.74	1.87
50%	3.23	1.54	1.91	1.53	2.33	3.79	8.08
75%	3.84	1.93	2.41	2.07	2.80	6.01	10.31
max	4.81	2.27	2.90	2.69	3.75	13.38	16.84

1.3 4.3 Define a fucntion and plot Data

[14]:
```
# Define a function using Plotly Express
def plotly_data(df, title):

    # Create figure
    fig = go.Figure()

    # Set title
    fig.update_layout(title_text = title)

    # For loop that plots all stock prices in the pandas dataframe df

    for i in df.columns[0:]:
        # Add range slider
        #fig.update_layout(xaxis=dict(rangeselector =␣
↪dict(buttons=list([dict(count=1, label="1m", step="month",␣
↪stepmode="backward"), dict(count=6, label="6m", step="month",␣
↪stepmode="backward"), dict(count=1, label="YTD", step="year",␣
↪stepmode="todate"), dict(count=1, label="1y", step="year",␣
↪stepmode="backward"), dict(step="all")])), rangeslider=dict( visible=True),␣
↪type="date"))
        # Add line graph
```

```
        fig.add_scatter(x = df.index, y = df[i], name = i)
        # Update Layout
        fig.update_layout({'plot_bgcolor': "white"})

    fig.show()

# Define a function using Plotly Express, changes axis y to logarithm scale
def log_plotly_data(df, title):

    # Create figure
    fig = go.Figure()

    # Set title
    fig.update_layout(title_text = title)

    # For loop that plots all stock prices in the pandas dataframe df

    for i in df.columns[0:]:
        # Add range slider
        #fig.update_layout(xaxis=dict(rangeselector =␣
 ↪dict(buttons=list([dict(count=1, label="1m", step="month",␣
 ↪stepmode="backward"), dict(count=6, label="6m", step="month",␣
 ↪stepmode="backward"), dict(count=1, label="YTD", step="year",␣
 ↪stepmode="todate"), dict(count=1, label="1y", step="year",␣
 ↪stepmode="backward"), dict(step="all")])), rangeslider=dict( visible=True),␣
 ↪type="date"))
        # Add line graph
        fig.add_scatter(x = df.index, y = df[i], name = i)
        # Update Layout
        fig.update_layout({'plot_bgcolor': "white"})

    #changes y to logarithm scale
    fig.update_yaxes(type="log")
    fig.show()
```

[15]:
```
plotly_data(Adj_Close_df, 'Closing Prices [$]')
```

[16]:
```
log_plotly_data(Adj_Close_df, 'Closing Prices [$]')
```

[17]:
```
plotly_data(scaling_df, "Scaling Closing Prices")
```

[18]:
```
log_plotly_data(scaling_df, "Scaling Closing Prices")
```

[19]:
```
plotly_data(p_change_df, "Percentage Daily Returns [%]")
```

[20]:
```
plotly_data(volume_df, "Trade Volume")
```