

If you do not have `pandas_datareader`, please install by running the following code.

```
In [1]: 1 !pip install pandas_datareader
```

We will import 'get\_data\_yahoo' from 'pandas\_datareader' data module.

```
In [2]: 1 import numpy as np
2 import pandas as pd
3 from datetime import datetime
4 from pandas_datareader.data import get_data_yahoo
```

The `get_data_yahoo()` method will fetch historical OHLC stock data from Yahoo Finance by parsing the web page. We specify the ticker and time range. The `now()` method from 'datetime' will return the current local date and time.

`get_data_yahoo()` requires tickers, start date, and end date. As an end date, we used the current date by using 'datetime.now()'

```
In [3]: 1 test = get_data_yahoo('QQQ', '2010-1-1', datetime.now())
2 test
```

Out[3]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2009-12-31	46.279999	45.750000	46.259998	45.750000	50079200.0	40.970905
2010-01-04	46.490002	46.270000	46.330002	46.419998	62822800.0	41.570904
2010-01-05	46.500000	46.160000	46.389999	46.419998	62935600.0	41.570904
2010-01-06	46.549999	46.070000	46.400002	46.139999	96033000.0	41.320183
2010-01-07	46.270000	45.919998	46.209999	46.169998	77094100.0	41.347046
...	...	...	...	...	...	...
2021-03-09	313.730011	306.920013	307.470001	311.769989	80107600.0	311.769989
2021-03-10	316.470001	310.170013	316.160004	310.880005	76547300.0	310.880005
2021-03-11	319.859985	314.850006	315.769989	318.040009	53244500.0	318.040009
2021-03-12	318.230011	311.390015	313.859985	315.459991	69276600.0	315.459991

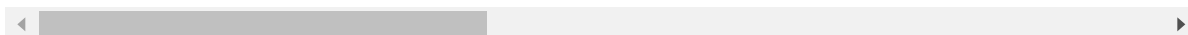
'get\_data\_yahoo' produce dataframe as below. Pay attention to the structure of the dataframe. It has all 'Adj Close' first for each ticker.

```
In [6]: 1 tickers = ["IVV", "IDEV", "IUSB", "IEMG", "IAGG", "IJH", "IJR"]
2 global_etf = get_data_yahoo(tickers, '2012-1-1', '2019-12-31')
3 global_etf
```

Out[6]:

Attributes	Adj Close							Clo
Symbols	IVV	IDEV	IUSB	IEMG	IAGG	IJH	IJR	IVV
Date								
2012-01-03	107.167709	NaN	NaN	NaN	NaN	77.248146	30.605028	128
2012-01-04	107.259796	NaN	NaN	NaN	NaN	77.030052	30.428633	128
2012-01-05	107.611389	NaN	NaN	NaN	NaN	77.623268	30.609446	128
2012-01-06	107.377014	NaN	NaN	NaN	NaN	77.562180	30.508009	128
2012-01-09	107.510925	NaN	NaN	NaN	NaN	77.989662	30.631491	128
...	...	...	...	...	...	...	...	...
2019-12-24	316.480499	56.514118	50.483654	52.225918	53.868710	202.210495	82.871719	322
2019-12-26	318.118591	56.915134	50.522446	52.637142	53.829487	202.652832	82.763382	324
2019-12-27	318.059723	56.983601	50.600044	52.862343	53.878517	202.318634	82.389114	324
2019-12-30	316.343201	56.699955	50.609741	52.490276	53.760860	202.122009	82.339859	322
2019-12-31	317.059235	56.895569	50.541851	52.637142	53.721634	202.318634	82.586090	323

2012 rows × 42 columns



We use `pct_change()` method to get daily return.

```
In [7]: 1 hist_return = global_etf['Adj Close'].pct_change()
        2 hist_return
```

Out[7]:

Symbols	IVV	IDEV	IUSB	IEMG	IAGG	IJH	IJR
Date							
2012-01-03	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-04	0.000859	NaN	NaN	NaN	NaN	-0.002823	-0.005764
2012-01-05	0.003278	NaN	NaN	NaN	NaN	0.007701	0.005942
2012-01-06	-0.002178	NaN	NaN	NaN	NaN	-0.000787	-0.003314
2012-01-09	0.001247	NaN	NaN	NaN	NaN	0.005511	0.004048
...	...	...	...	...	...	...	...
2019-12-24	0.000124	-0.003621	0.000769	-0.001684	0.001002	-0.000097	0.002741
2019-12-26	0.005176	0.007096	0.000768	0.007874	-0.000728	0.002188	-0.001307
2019-12-27	-0.000185	0.001203	0.001536	0.004278	0.000911	-0.001649	-0.004522
2019-12-30	-0.005397	-0.004978	0.000192	-0.007038	-0.002184	-0.000972	-0.000598

DataFrame has a `cov()` method that will return the DataFrame of covariance of columns.

```
In [9]: 1 hist_return_cov = hist_return.cov()
        2 hist_return_cov
```

Out[9]:

Symbols	IVV	IDEV	IUSB	IEMG	IAGG	IJH	IJR
Symbols							
IVV	0.000065	0.000050	-2.203952e-06	6.715897e-05	-0.000002	0.000066	0.000068
IDEV	0.000050	0.000053	-1.387950e-06	5.999208e-05	-0.000002	0.000049	0.000051
IUSB	-0.000002	-0.000001	3.987899e-06	-8.202842e-07	0.000002	-0.000003	-0.000003
IEMG	0.000067	0.000060	-8.202842e-07	1.207803e-04	-0.000001	0.000069	0.000070
IAGG	-0.000002	-0.000002	1.992607e-06	-1.082100e-06	0.000005	-0.000003	-0.000004
IJH	0.000066	0.000049	-2.597844e-06	6.880446e-05	-0.000003	0.000080	0.000084
IJR	0.000068	0.000051	-3.410843e-06	7.036246e-05	-0.000004	0.000084	0.000098

The `corr()` method will return correlation of columns.

```
In [10]: 1 hist_return_corr = hist_return.corr()
        2 hist_return_corr
```

Out[10]:

Symbols	IVV	IDEV	IUSB	IEMG	IAGG	IJH	IJR
Symbols							
IVV	1.000000	0.829442	-0.132103	0.755313	-0.127700	0.920529	0.858642
IDEV	0.829442	1.000000	-0.115620	0.796034	-0.099467	0.771027	0.712270
IUSB	-0.132103	-0.115620	1.000000	-0.036657	0.503555	-0.145410	-0.170278
IEMG	0.755313	0.796034	-0.036657	1.000000	-0.044829	0.709004	0.650524
IAGG	-0.127700	-0.099467	0.503555	-0.044829	1.000000	-0.144412	-0.166088
IJH	0.920529	0.771027	-0.145410	0.709004	-0.144412	1.000000	0.950489
IJR	0.858642	0.712270	-0.170278	0.650524	-0.166088	0.950489	1.000000

'np's 'diag' method will take the elements of the matrix in diagonal.

```
In [11]: 1 annual_sd = pd.DataFrame(np.sqrt(np.diag(hist_return_cov)) * np.sqrt(250))
        2 annual_sd
```

Out[11]:


	0
IVV	0.127629
IDEV	0.115259
IUSB	0.031575
IEMG	0.173767
IAGG	0.034613
IJH	0.141100
IJR	0.156187

np.std() will return the standard deviation of specified column. 'dropna()' will drop missing values from the data.

```
In [12]: 1 np.std(hist_return['IDEV'].dropna()) * np.sqrt(250)
```

Out[12]: 0.11517618457720198

Save historical return data.

In [13]:  1 hist\_return.to\_csv('hist\_return.csv')

In [ ]:  1