02_yfinance_demo

September 29, 2021

1 Downloading Market and Fundamental Data with `yfinance`

1.1 Imports & Settings

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import pandas as pd
import yfinance as yf
```

1.2 How to work with a Ticker object

[3]:
```
symbol = 'FB'
ticker = yf.Ticker(symbol)
```

1.2.1 Show ticker info

[4]:
```
pd.Series(ticker.info).head(20)
```

[4]:
```
zip                                                                94025
sector                                                Communication Services
fullTimeEmployees                                                  56653
longBusinessSummary           Facebook, Inc. develops products that enable p…
city                                                          Menlo Park
phone                                                       650-543-4800
state                                                                 CA
country                                                   United States
companyOfficers                                                       []
website                                               http://investor.fb.com
maxAge                                                                 1
address1                                                1601 Willow Road
industry                                  Internet Content & Information
previousClose                                                   276.78000
regularMarketOpen                                               277.95000
twoHundredDayAverage                                            263.02948
trailingAnnualDividendYield                                          None
payoutRatio                                                             0
```

```
volume24Hr                                                    None
regularMarketDayHigh                                      278.07500
dtype: object
```

1.2.2 Get market data

[5]:
```
data = ticker.history(period='5d',
                      interval='1m',
                      start=None,
                      end=None,
                      actions=True,
                      auto_adjust=True,
                      back_adjust=False)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1771 entries, 2020-12-23 09:30:00-05:00 to 2020-12-30
15:59:00-05:00
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Open          1771 non-null   float64
 1   High          1771 non-null   float64
 2   Low           1771 non-null   float64
 3   Close         1771 non-null   float64
 4   Volume        1771 non-null   int64
 5   Dividends     1771 non-null   int64
 6   Stock Splits  1771 non-null   int64
dtypes: float64(4), int64(3)
memory usage: 110.7 KB
```

1.2.3 View company actions

[6]:
```
# show actions (dividends, splits)
ticker.actions
```

[6]:
```
Empty DataFrame
Columns: [Dividends, Stock Splits]
Index: []
```

[7]:
```
ticker.dividends
```

[7]:
```
Series([], Name: Dividends, dtype: int64)
```

[8]:
```
ticker.splits
```

[8]:
```
Series([], Name: Stock Splits, dtype: int64)
```

1.2.4 Annual and Quarterly Financial Statement Summary

[9]:
```
ticker.financials
```

[9]:
```
Empty DataFrame
Columns: [Open, High, Low, Close, Adj Close, Volume]
Index: []
```

[10]:
```
ticker.quarterly_financials
```

[10]:
```
Empty DataFrame
Columns: [Open, High, Low, Close, Adj Close, Volume]
Index: []
```

1.2.5 Annual and Quarterly Balance Sheet

[11]:
```
ticker.balance_sheet
```

[11]:
```
Empty DataFrame
Columns: [Open, High, Low, Close, Adj Close, Volume]
Index: []
```

[12]:
```
ticker.quarterly_balance_sheet
```

[12]:
```
Empty DataFrame
Columns: [Open, High, Low, Close, Adj Close, Volume]
Index: []
```

1.2.6 Annual and Quarterly Cashflow Statement

[13]:
```
ticker.cashflow
```

[13]:
```
Empty DataFrame
Columns: [Open, High, Low, Close, Adj Close, Volume]
Index: []
```

[14]:
```
ticker.quarterly_cashflow
```

[14]:
```
Empty DataFrame
Columns: [Open, High, Low, Close, Adj Close, Volume]
Index: []
```

[15]:
```
ticker.earnings
```

[15]:
```
Empty DataFrame
Columns: [Open, High, Low, Close, Adj Close, Volume]
Index: []
```

[16]:
```
ticker.quarterly_earnings
```

[16]:
```
Empty DataFrame
Columns: [Open, High, Low, Close, Adj Close, Volume]
Index: []
```

1.2.7 Sustainability: Environmental, Social and Governance (ESG)

[17]:
```
ticker.sustainability
```

[17]:

	Value
2020-10	
palmOil	False
controversialWeapons	False
gambling	False
socialScore	17.68
nuclear	False
furLeather	False
alcoholic	False
gmo	False
catholic	False
socialPercentile	None
peerCount	102
governanceScore	12.28
environmentPercentile	None
animalTesting	False
tobacco	False
totalEsg	31.40
highestControversy	4
esgPerformance	OUT_PERF
coal	False
pesticides	False
adult	False
percentile	61.39
peerGroup	Software & Services
smallArms	False
environmentScore	1.43
governancePercentile	None
militaryContract	False

1.2.8 Analyst Recommendations

[18]:
```
ticker.recommendations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 511 entries, 2012-06-22 07:56:00 to 2020-10-30 17:14:48
Data columns (total 4 columns):
```

```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Firm        511 non-null    object
 1   To Grade    511 non-null    object
 2   From Grade  511 non-null    object
 3   Action      511 non-null    object
dtypes: object(4)
memory usage: 20.0+ KB
```

[19]:
```
ticker.recommendations.tail(10)
```

[19]:

Date	Firm	To Grade	From Grade	Action
2020-10-20 12:31:45	Credit Suisse	Outperform		main
2020-10-26 08:50:44	KeyBanc	Overweight		main
2020-10-28 16:47:21	JP Morgan	Overweight		main
2020-10-30 11:53:22	Raymond James	Strong Buy		main
2020-10-30 12:45:43	Credit Suisse	Outperform		main
2020-10-30 12:50:35	Morgan Stanley	Overweight		main
2020-10-30 15:15:15	Canaccord Genuity	Buy		main
2020-10-30 17:03:14	Mizuho	Buy		main
2020-10-30 17:06:53	Wells Fargo	Overweight		main
2020-10-30 17:14:48	Truist Securities	Buy		main

1.2.9 Upcoming Events

[20]:
```
ticker.calendar
```

[20]:

	0	1
Earnings Date	2021-01-27 00:00:00	2021-02-01 00:00:00
Earnings Average	3.19	3.19
Earnings Low	2.67	2.67
Earnings High	3.63	3.63
Revenue Average	26288700000	26288700000
Revenue Low	25412000000	25412000000
Revenue High	26993000000	26993000000

1.2.10 Option Expiration Dates

[21]:
```
ticker.options
```

[21]:
```
('2020-12-30',
 '2021-01-07',
 '2021-01-14',
 '2021-01-21',
 '2021-01-28',
 '2021-02-04',
```

```
 '2021-02-18',
 '2021-03-18',
 '2021-06-17',
 '2021-07-15',
 '2021-09-16',
 '2022-01-20',
 '2022-06-16',
 '2022-09-15',
 '2023-01-19',
 '2026-02-20')
```

[22]:
```
expiration = ticker.options[0]
```

[23]:
```
options = ticker.option_chain(expiration)
```

[24]:
```
options.calls.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   contractSymbol     45 non-null     object
 1   lastTradeDate      45 non-null     datetime64[ns]
 2   strike             45 non-null     float64
 3   lastPrice          45 non-null     float64
 4   bid                45 non-null     float64
 5   ask                45 non-null     float64
 6   change             45 non-null     float64
 7   percentChange      45 non-null     float64
 8   volume             45 non-null     int64
 9   openInterest       45 non-null     int64
 10  impliedVolatility  45 non-null     float64
 11  inTheMoney         45 non-null     bool
 12  contractSize       45 non-null     object
 13  currency           45 non-null     object
dtypes: bool(1), datetime64[ns](1), float64(7), int64(2), object(3)
memory usage: 4.7+ KB
```

[25]:
```
options.calls.head()
```

[25]:

	contractSymbol	lastTradeDate	strike	lastPrice	bid	ask	\
0	FB201231C00195000	2020-12-30 20:54:40	195.0	77.60	76.35	77.5	
1	FB201231C00200000	2020-12-30 20:39:19	200.0	72.50	71.35	72.5	
2	FB201231C00205000	2020-12-30 20:22:11	205.0	68.29	66.35	67.5	
3	FB201231C00210000	2020-12-29 20:31:41	210.0	68.18	61.35	62.5	
4	FB201231C00215000	2020-12-24 15:29:26	215.0	55.50	56.35	57.5	

```
     change  percentChange  volume  openInterest  impliedVolatility  \
0 -4.200005      -5.134480       1            21           1.773439
1 -5.250000      -6.752411       7            33           1.648439
2 -4.870003      -6.656647      11            49           1.531252
3  0.000000       0.000000       2            43           1.414065
4  0.000000       0.000000       3            35           1.296879

   inTheMoney contractSize currency
0        True      REGULAR      USD
1        True      REGULAR      USD
2        True      REGULAR      USD
3        True      REGULAR      USD
4        True      REGULAR      USD
```

[26]:
```
options.puts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43 entries, 0 to 42
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   contractSymbol     43 non-null     object
 1   lastTradeDate      43 non-null     datetime64[ns]
 2   strike             43 non-null     float64
 3   lastPrice          43 non-null     float64
 4   bid                43 non-null     float64
 5   ask                43 non-null     float64
 6   change             43 non-null     float64
 7   percentChange      43 non-null     float64
 8   volume             42 non-null     float64
 9   openInterest       43 non-null     int64
 10  impliedVolatility  43 non-null     float64
 11  inTheMoney         43 non-null     bool
 12  contractSize       43 non-null     object
 13  currency           43 non-null     object
dtypes: bool(1), datetime64[ns](1), float64(8), int64(1), object(3)
memory usage: 4.5+ KB
```

1.3 Data Download with proxy server

You can use a proxy server to avoid having your IP blacklisted as illustrated below (but need an actual PROXY_SERVER).

[27]:
```
PROXY_SERVER = 'PROXY_SERVER'
```

The following will only work with proper PROXY_SERVER...

[28]:
```
# msft = yf.Ticker("MSFT")

# msft.history(proxy=PROXY_SERVER)
# msft.get_actions(proxy=PROXY_SERVER)
# msft.get_dividends(proxy=PROXY_SERVER)
# msft.get_splits(proxy=PROXY_SERVER)
# msft.get_balance_sheet(proxy=PROXY_SERVER)
# msft.get_cashflow(proxy=PROXY_SERVER)
# msgt.option_chain(proxy=PROXY_SERVER)
```

1.4 Downloading multiple symbols

[29]:
```
tickers = yf.Tickers('msft aapl goog')
```

[30]:
```
tickers
```

[30]:
```
yfinance.Tickers object <MSFT,AAPL,GOOG>
```

[31]:
```
pd.Series(tickers.tickers['MSFT'].info)
```

[31]:
```
zip                                                        98052-6399
sector                                                     Technology
fullTimeEmployees                                              163000
longBusinessSummary        Microsoft Corporation develops, licenses, and …
city                                                         Redmond
                                                   …
impliedSharesOutstanding                                          None
category                                                          None
fiveYearAverageReturn                                             None
regularMarketPrice                                              225.23
logo_url                                https://logo.clearbit.com/microsoft.com
Length: 123, dtype: object
```

[32]:
```
tickers.tickers['AAPL'].history(period="1mo")
```

[32]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2020-11-30	116.97	120.97	116.81	119.05	169410200	0	0
2020-12-01	121.01	123.47	120.01	122.72	128166800	0	0
2020-12-02	122.02	123.37	120.89	123.08	89004200	0	0
2020-12-03	123.52	123.78	122.21	122.94	78967600	0	0
2020-12-04	122.60	122.86	121.52	122.25	78260400	0	0
2020-12-07	122.31	124.57	122.25	123.75	86712000	0	0
2020-12-08	124.37	124.98	123.09	124.38	82225500	0	0
2020-12-09	124.53	125.95	121.00	121.78	115089200	0	0
2020-12-10	120.50	123.87	120.15	123.24	81312200	0	0
2020-12-11	122.43	122.76	120.55	122.41	86939800	0	0

```
2020-12-14  122.60  123.35  121.54  121.78   79184500          0             0
2020-12-15  124.34  127.90  124.13  127.88  157572300          0             0
2020-12-16  127.41  128.37  126.56  127.81   98208600          0             0
2020-12-17  128.90  129.58  128.04  128.70   94359800          0             0
2020-12-18  128.96  129.10  126.12  126.66  192541500          0             0
2020-12-21  125.02  128.31  123.45  128.23  121251600          0             0
2020-12-22  131.61  134.41  129.65  131.88  168904800          0             0
2020-12-23  132.16  132.43  130.78  130.96   88223700          0             0
2020-12-24  131.32  133.46  131.10  131.97   54930100          0             0
2020-12-28  133.99  137.34  133.51  136.69  124486200          0             0
2020-12-29  138.05  138.79  134.34  134.87  120778200          0             0
2020-12-30  135.58  135.99  133.40  133.72   92882124          0             0
```

[33]:
```
tickers.history(period='1mo').stack(-1)
```

```
[**********************100%**********************]  3 of 3 completed
```

[33]:
```
                        Close  Dividends         High          Low  \
Date
2020-11-30 AAPL    119.050003          0   120.970001   116.809998
           GOOG   1760.739990          0  1788.064941  1755.000000
           MSFT    214.070007          0   214.759995   210.839996
2020-12-01 AAPL    122.720001          0   123.470001   120.010002
           GOOG   1798.099976          0  1824.829956  1769.369995
…                         …        …           …            …
2020-12-29 GOOG   1758.719971          0  1792.439941  1756.089966
           MSFT    224.149994          0   227.179993   223.580002
2020-12-30 AAPL    133.720001          0   135.990005   133.399994
           GOOG   1739.520020          0  1765.094971  1725.680054
           MSFT    221.679993          0   225.630005   221.470001

                         Open  Stock Splits     Volume
Date
2020-11-30 AAPL    116.970001             0  169410200
           GOOG   1781.183960             0    1823800
           MSFT    214.100006             0   33064800
2020-12-01 AAPL    121.010002             0  128166800
           GOOG   1774.369995             0    1739000
…                         …           …         …
2020-12-29 GOOG   1787.790039             0    1298600
           MSFT    226.309998             0   17348000
2020-12-30 AAPL    135.580002             0   92882124
           GOOG   1762.010010             0    1293285
           MSFT    225.229996             0   19273172

[66 rows x 7 columns]
```

[34]:
```
data = yf.download("SPY AAPL", start="2020-01-01", end="2020-01-05")
```

```
[*********************100%***********************]  2 of 2 completed
```

[35]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2 entries, 2020-01-02 to 2020-01-03
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   (Adj Close, AAPL)  2 non-null      float64
 1   (Adj Close, SPY)   2 non-null      float64
 2   (Close, AAPL)      2 non-null      float64
 3   (Close, SPY)       2 non-null      float64
 4   (High, AAPL)       2 non-null      float64
 5   (High, SPY)        2 non-null      float64
 6   (Low, AAPL)        2 non-null      float64
 7   (Low, SPY)         2 non-null      float64
 8   (Open, AAPL)       2 non-null      float64
 9   (Open, SPY)        2 non-null      float64
 10  (Volume, AAPL)     2 non-null      int64  
 11  (Volume, SPY)      2 non-null      int64  
dtypes: float64(10), int64(2)
memory usage: 208.0 bytes
```

[36]:
```
data = yf.download(
        tickers = "SPY AAPL MSFT", # list or string

        # use "period" instead of start/end
        # valid periods: 1d,5d,1mo,3mo,6mo,1y,2y,5y,10y,ytd,max
        # (optional, default is '1mo')
        period = "5d",

        # fetch data by interval (including intraday if period < 60 days)
        # valid intervals: 1m,2m,5m,15m,30m,60m,90m,1h,1d,5d,1wk,1mo,3mo
        # (optional, default is '1d')
        interval = "1m",

        # group by ticker (to access via data['SPY'])
        # (optional, default is 'column')
        group_by = 'ticker',

        # adjust all OHLC automatically
        # (optional, default is False)
        auto_adjust = True,
```

```
        # download pre/post regular market hours data
        # (optional, default is False)
        prepost = True,

        # use threads for mass downloading? (True/False/Integer)
        # (optional, default is True)
        threads = True,

        # proxy URL scheme use use when downloading?
        # (optional, default is None)
        proxy = None
    )
```

```
[**********100%**********]  3 of 3 completed
```

[37]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 4045 entries, 2020-12-23 04:15:00-05:00 to 2020-12-30
17:10:00-05:00
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   (MSFT, Open)    2500 non-null   float64
 1   (MSFT, High)    2500 non-null   float64
 2   (MSFT, Low)     2500 non-null   float64
 3   (MSFT, Close)   2500 non-null   float64
 4   (MSFT, Volume)  2500 non-null   float64
 5   (AAPL, Open)    3843 non-null   float64
 6   (AAPL, High)    3843 non-null   float64
 7   (AAPL, Low)     3843 non-null   float64
 8   (AAPL, Close)   3843 non-null   float64
 9   (AAPL, Volume)  3843 non-null   float64
 10  (SPY, Open)     3358 non-null   float64
 11  (SPY, High)     3358 non-null   float64
 12  (SPY, Low)      3358 non-null   float64
 13  (SPY, Close)    3358 non-null   float64
 14  (SPY, Volume)   3358 non-null   float64
dtypes: float64(15)
memory usage: 505.6 KB
```

[38]:
```
from pandas_datareader import data as pdr

import yfinance as yf
yf.pdr_override()

# download dataframe
```

```
data = pdr.get_data_yahoo('SPY',
                          start='2017-01-01',
                          end='2019-04-30',
                          auto_adjust=False)
```

```
[**********************100%**********************]  1 of 1 completed
```

[39]:
```
# auto_adjust = True
data.tail()
```

[39]:

Date	Open	High	Low	Close	Adj Close	Volume
2019-04-23	290.679993	293.140015	290.420013	292.880005	283.410553	52246600
2019-04-24	292.790009	293.160004	292.070007	292.230011	282.781616	50392900
2019-04-25	292.119995	292.779999	290.730011	292.049988	282.607452	57770900
2019-04-26	292.100006	293.489990	291.239990	293.410004	283.923462	50916400
2019-04-29	293.510010	294.450012	293.410004	293.869995	284.368561	57197700

[40]:
```
# auto_adjust = False
data.tail()
```

[40]:

Date	Open	High	Low	Close	Adj Close	Volume
2019-04-23	290.679993	293.140015	290.420013	292.880005	283.410553	52246600
2019-04-24	292.790009	293.160004	292.070007	292.230011	282.781616	50392900
2019-04-25	292.119995	292.779999	290.730011	292.049988	282.607452	57770900
2019-04-26	292.100006	293.489990	291.239990	293.410004	283.923462	50916400
2019-04-29	293.510010	294.450012	293.410004	293.869995	284.368561	57197700

[ ]:
```

```