01_pandas_datareader_demo

September 29, 2021

1 Remote data access using pandas

The pandas library enables access to data displayed on websites using the `read_html()` function and access to the API endpoints of various data providers through the related `pandas-datareader` library.

1.1 Imports & Settings

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

[2]:
```
%matplotlib inline
import os
from datetime import datetime
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import mplfinance as mpf
import seaborn as sns
```

1.2 Download html table with SP500 constituents

The download of the content of one or more html tables works as follows, for instance for the constituents of the S&P500 index from Wikipedia

[3]:
```
sp_url = 'https://en.wikipedia.org/wiki/List_of_S%26P_500_companies'
sp500_constituents = pd.read_html(sp_url, header=0)[0]
```

[4]:
```
sp500_constituents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 505 entries, 0 to 504
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Symbol                 505 non-null    object
 1   Security               505 non-null    object
 2   SEC filings            505 non-null    object
```

```
 3   GICS Sector            505 non-null    object
 4   GICS Sub-Industry      505 non-null    object
 5   Headquarters Location  505 non-null    object
 6   Date first added       453 non-null    object
 7   CIK                    505 non-null    int64
 8   Founded                505 non-null    object
dtypes: int64(1), object(8)
memory usage: 35.6+ KB
```

[5]:
```
sp500_constituents.head()
```

[5]:

	Symbol	Security	SEC filings	GICS Sector	GICS Sub-Industry	Headquarters Location	Date first added	CIK	Founded
0	MMM	3M Company	reports	Industrials	Industrial Conglomerates	St. Paul, Minnesota	1976-08-09	66740	1902
1	ABT	Abbott Laboratories	reports	Health Care	Health Care Equipment	North Chicago, Illinois	1964-03-31	1800	1888
2	ABBV	AbbVie Inc.	reports	Health Care	Pharmaceuticals	North Chicago, Illinois	2012-12-31	1551152	2013 (1888)
3	ABMD	Abiomed	reports	Health Care	Health Care Equipment	Danvers, Massachusetts	2018-05-31	815094	1981
4	ACN	Accenture	reports	Information Technology	IT Consulting & Other Services	Dublin, Ireland	2011-07-06	1467373	1989

1.3 pandas-datareader for Market Data

`pandas` used to facilitate access to data providers' APIs directly, but this functionality has moved to the related pandas-datareader library. The stability of the APIs varies with provider policies, and as of June 2o18 at version 0.7, the following sources are available

See documentation; functionality frequently changes as underlying provider APIs evolve.

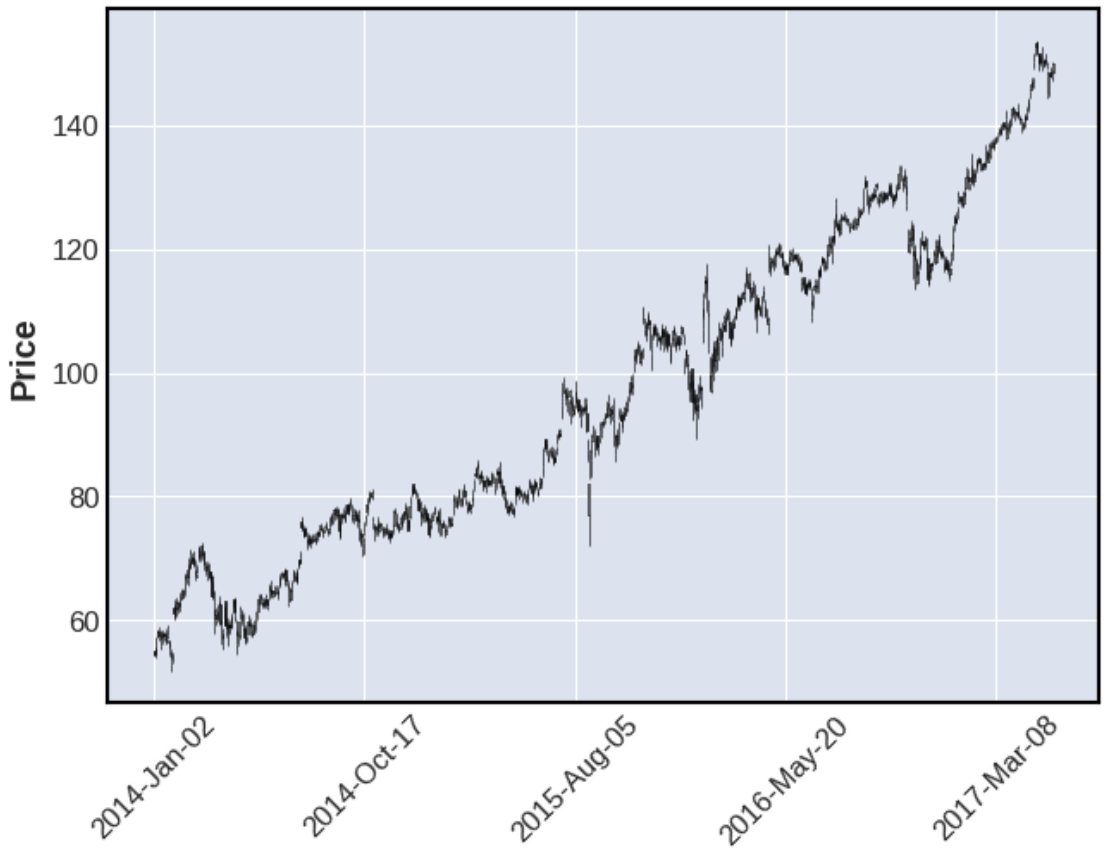
1.3.1 Yahoo Finance

[6]:
```
start = '2014'
end = datetime(2017, 5, 24)

yahoo= web.DataReader('FB', 'yahoo', start=start, end=end)
yahoo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 855 entries, 2014-01-02 to 2017-05-24
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   High       855 non-null    float64
 1   Low        855 non-null    float64
 2   Open       855 non-null    float64
 3   Close      855 non-null    float64
 4   Volume     855 non-null    int64
 5   Adj Close  855 non-null    float64
dtypes: float64(5), int64(1)
memory usage: 46.8 KB
```

[7]:
```
mpf.plot(yahoo.drop('Adj Close', axis=1), type='candle')
plt.tight_layout()
```




```
<Figure size 640x480 with 0 Axes>
```

1.3.2 IEX

IEX is an alternative exchange started in response to the HFT controversy and portrayed in Michael Lewis' controversial Flash Boys. It aims to slow down the speed of trading to create a more level playing field and has been growing rapidly since launch in 2016 while still small with a market share of around 2.5% in June 2018.

> **Note:** IEX now requires an API key after registration for (free) account that you can store as environment variable and retrieve as illustrated below, or pass directly via keyword argument to `pandas_datareader`.

[8]:
```
IEX_API_KEY=os.getenv('IEX_API_KEY')
```

[9]:
```
start = datetime(2015, 2, 9)
# end = datetime(2017, 5, 24)

iex = web.DataReader('FB', 'iex', start, api_key=IEX_API_KEY)
iex.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1519 entries, 2015-02-09 to 2021-02-19
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   open    1519 non-null   float64
 1   high    1519 non-null   float64
 2   low     1519 non-null   float64
 3   close   1519 non-null   float64
 4   volume  1519 non-null   int64  
dtypes: float64(4), int64(1)
memory usage: 71.2+ KB
```

[10]:
```
iex.tail()
```

[10]:

date	open	high	low	close	volume
2021-02-12	270.520	271.18	268.34	270.50	9097597
2021-02-16	270.800	276.60	270.05	273.97	15417243
2021-02-17	271.240	273.97	269.58	273.57	12763240
2021-02-18	269.565	271.95	266.03	269.39	15249134
2021-02-19	269.860	270.27	260.15	261.56	25622587

[11]:
```
sns.set_style('whitegrid')
iex.close.plot(figsize=(14, 5))
sns.despine()
```

Book Data In addition to historical EOD price and volume data, IEX provides real-time depth of book quotations that offer an aggregated size of orders by price and side. This service also includes last trade price and size information.

DEEP is used to receive real-time depth of book quotations direct from IEX. The depth of book quotations received via DEEP provide an aggregated size of resting displayed orders at a price and side, and do not indicate the size or number of individual orders at any price level. Non-displayed orders and non-displayed portions of reserve orders are not represented in DEEP.

DEEP also provides last trade price and size information. Trades resulting from either displayed or non-displayed orders matching on IEX will be reported. Routed executions will not be reported.

Only works on trading days.

[12]:
```
book = web.get_iex_book('AAPL')
```

[13]:
```
list(book.keys())
```

[13]:
```
['symbol',
 'marketPercent',
 'volume',
 'lastSalePrice',
 'lastSaleSize',
 'lastSaleTime',
 'lastUpdated',
 'bids',
 'asks',
 'systemEvent',
 'tradingStatus',
 'opHaltStatus',
 'ssrStatus',
 'securityEvent',
 'trades',
```

```
 'tradeBreaks']
```

[14]:
```
orders = pd.concat([pd.DataFrame(book[side]).assign(side=side) for side in␣
↪['bids', 'asks']])
orders.head()
```

[14]:
```
Empty DataFrame
Columns: [side]
Index: []
```

[15]:
```
for key in book.keys():
    try:
        print(f'\n{key}')
        print(pd.DataFrame(book[key]))
    except:
        print(book[key])
```

```
symbol
AAPL

marketPercent
0.01824

volume
1874997

lastSalePrice
125.98

lastSaleSize
3

lastSaleTime
1614027994379

lastUpdated
1614031191208

bids
Empty DataFrame
Columns: []
Index: []

asks
Empty DataFrame
Columns: []
```

```
Index: []

systemEvent
{'systemEvent': 'C', 'timestamp': 1614031800007}

tradingStatus
{'status': 'T', 'reason': '    ', 'timestamp': 1613996038606}

opHaltStatus
{'isHalted': False, 'timestamp': 1613996038606}

ssrStatus
{'isSSR': False, 'detail': ' ', 'timestamp': 1613996038606}

securityEvent
{'securityEvent': 'MarketClose', 'timestamp': 1614027600000}

trades
      price  size     tradeId  isISO  isOddLot  isOutsideRegularHours  \
0   125.980     3  2565301038   True      True                   True
1   126.010    20  2561192133  False      True                  False
2   126.010    35  2561162510  False      True                  False
3   126.005   100  2560819178  False     False                  False
4   126.005   100  2560535358  False     False                  False
5   126.010   100  2559785204  False     False                  False
6   126.020   132  2559650792  False     False                  False
7   126.045   100  2559329974  False     False                  False
8   126.060   300  2559317473   True     False                  False
9   126.050     1  2559295066   True      True                  False
10  126.035   100  2559185683  False     False                  False
11  126.010   300  2558525991  False     False                  False
12  126.010     6  2558455856   True      True                  False
13  126.010    30  2558411929   True      True                  False
14  126.020   300  2558025659  False     False                  False
15  126.010   300  2558014028  False     False                  False
16  126.015   100  2557970786  False     False                  False
17  126.020    90  2557488823   True      True                  False
18  126.020     3  2557462987   True      True                  False
19  126.000   100  2557328682   True     False                  False

    isSinglePriceCross  isTradeThroughExempt      timestamp
0                False                 False  1614027994379
1                False                 False  1614027597753
2                False                 False  1614027597658
3                False                 False  1614027596970
4                False                 False  1614027596307
5                False                 False  1614027594703
6                False                 False  1614027594402
```

```
7                False                False  1614027593980
8                False                 True  1614027593950
9                False                False  1614027593916
10               False                False  1614027593636
11               False                False  1614027592039
12               False                False  1614027591885
13               False                False  1614027591777
14               False                False  1614027591036
15               False                False  1614027591023
16               False                False  1614027590959
17               False                False  1614027590145
18               False                False  1614027590119
19               False                False  1614027590005

tradeBreaks
Empty DataFrame
Columns: []
Index: []
```

[16]:
```
pd.DataFrame(book['trades']).head()
```

[16]:

	price	size	tradeId	isISO	isOddLot	isOutsideRegularHours	isSinglePriceCross	isTradeThroughExempt	timestamp
0	125.980	3	2565301038	True	True	True	False	False	1614027994379
1	126.010	20	2561192133	False	True	False	False	False	1614027597753
2	126.010	35	2561162510	False	True	False	False	False	1614027597658
3	126.005	100	2560819178	False	False	False	False	False	1614027596970
4	126.005	100	2560535358	False	False	False	False	False	1614027596307

1.3.3 Quandl

Obtain Quandl API Key and store in environment variable as `QUANDL_API_KEY`.

[17]:
```
symbol = 'FB.US'

quandl = web.DataReader(symbol, 'quandl', '2015-01-01')
quandl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 813 entries, 2018-03-27 to 2015-01-02
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
```

```
---  ------      --------------  -----
 0   Open        813 non-null    float64
 1   High        813 non-null    float64
 2   Low         813 non-null    float64
 3   Close       813 non-null    float64
 4   Volume      813 non-null    float64
 5   ExDividend  813 non-null    float64
 6   SplitRatio  813 non-null    float64
 7   AdjOpen     813 non-null    float64
 8   AdjHigh     813 non-null    float64
 9   AdjLow      813 non-null    float64
 10  AdjClose    813 non-null    float64
 11  AdjVolume   813 non-null    float64
dtypes: float64(12)
memory usage: 82.6 KB
```

1.3.4 FRED

[18]:
```
start = datetime(2010, 1, 1)

end = datetime(2013, 1, 27)

gdp = web.DataReader('GDP', 'fred', start, end)

gdp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 13 entries, 2010-01-01 to 2013-01-01
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   GDP     13 non-null     float64
dtypes: float64(1)
memory usage: 208.0 bytes
```

[19]:
```
inflation = web.DataReader(['CPIAUCSL', 'CPILFESL'], 'fred', start, end)
inflation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 37 entries, 2010-01-01 to 2013-01-01
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   CPIAUCSL  37 non-null     float64
 1   CPILFESL  37 non-null     float64
dtypes: float64(2)
memory usage: 888.0 bytes
```

1.3.5 Fama/French

[20]:
```
from pandas_datareader.famafrench import get_available_datasets
get_available_datasets()
```

[20]:
```
['F-F_Research_Data_Factors',
 'F-F_Research_Data_Factors_weekly',
 'F-F_Research_Data_Factors_daily',
 'F-F_Research_Data_5_Factors_2x3',
 'F-F_Research_Data_5_Factors_2x3_daily',
 'Portfolios_Formed_on_ME',
 'Portfolios_Formed_on_ME_Wout_Div',
 'Portfolios_Formed_on_ME_Daily',
 'Portfolios_Formed_on_BE-ME',
 'Portfolios_Formed_on_BE-ME_Wout_Div',
 'Portfolios_Formed_on_BE-ME_Daily',
 'Portfolios_Formed_on_OP',
 'Portfolios_Formed_on_OP_Wout_Div',
 'Portfolios_Formed_on_OP_Daily',
 'Portfolios_Formed_on_INV',
 'Portfolios_Formed_on_INV_Wout_Div',
 'Portfolios_Formed_on_INV_Daily',
 '6_Portfolios_2x3',
 '6_Portfolios_2x3_Wout_Div',
 '6_Portfolios_2x3_weekly',
 '6_Portfolios_2x3_daily',
 '25_Portfolios_5x5',
 '25_Portfolios_5x5_Wout_Div',
 '25_Portfolios_5x5_Daily',
 '100_Portfolios_10x10',
 '100_Portfolios_10x10_Wout_Div',
 '100_Portfolios_10x10_Daily',
 '6_Portfolios_ME_OP_2x3',
 '6_Portfolios_ME_OP_2x3_Wout_Div',
 '6_Portfolios_ME_OP_2x3_daily',
 '25_Portfolios_ME_OP_5x5',
 '25_Portfolios_ME_OP_5x5_Wout_Div',
 '25_Portfolios_ME_OP_5x5_daily',
 '100_Portfolios_ME_OP_10x10',
 '100_Portfolios_10x10_ME_OP_Wout_Div',
 '100_Portfolios_ME_OP_10x10_daily',
 '6_Portfolios_ME_INV_2x3',
 '6_Portfolios_ME_INV_2x3_Wout_Div',
 '6_Portfolios_ME_INV_2x3_daily',
 '25_Portfolios_ME_INV_5x5',
 '25_Portfolios_ME_INV_5x5_Wout_Div',
 '25_Portfolios_ME_INV_5x5_daily',
```

```
'100_Portfolios_ME_INV_10x10',
'100_Portfolios_10x10_ME_INV_Wout_Div',
'100_Portfolios_ME_INV_10x10_daily',
'25_Portfolios_BEME_OP_5x5',
'25_Portfolios_BEME_OP_5x5_Wout_Div',
'25_Portfolios_BEME_OP_5x5_daily',
'25_Portfolios_BEME_INV_5x5',
'25_Portfolios_BEME_INV_5x5_Wout_Div',
'25_Portfolios_BEME_INV_5x5_daily',
'25_Portfolios_OP_INV_5x5',
'25_Portfolios_OP_INV_5x5_Wout_Div',
'25_Portfolios_OP_INV_5x5_daily',
'32_Portfolios_ME_BEME_OP_2x4x4',
'32_Portfolios_ME_BEME_OP_2x4x4_Wout_Div',
'32_Portfolios_ME_BEME_INV_2x4x4',
'32_Portfolios_ME_BEME_INV_2x4x4_Wout_Div',
'32_Portfolios_ME_OP_INV_2x4x4',
'32_Portfolios_ME_OP_INV_2x4x4_Wout_Div',
'Portfolios_Formed_on_E-P',
'Portfolios_Formed_on_E-P_Wout_Div',
'Portfolios_Formed_on_CF-P',
'Portfolios_Formed_on_CF-P_Wout_Div',
'Portfolios_Formed_on_D-P',
'Portfolios_Formed_on_D-P_Wout_Div',
'6_Portfolios_ME_EP_2x3',
'6_Portfolios_ME_EP_2x3_Wout_Div',
'6_Portfolios_ME_CFP_2x3',
'6_Portfolios_ME_CFP_2x3_Wout_Div',
'6_Portfolios_ME_DP_2x3',
'6_Portfolios_ME_DP_2x3_Wout_Div',
'F-F_Momentum_Factor',
'F-F_Momentum_Factor_daily',
'6_Portfolios_ME_Prior_12_2',
'6_Portfolios_ME_Prior_12_2_Daily',
'25_Portfolios_ME_Prior_12_2',
'25_Portfolios_ME_Prior_12_2_Daily',
'10_Portfolios_Prior_12_2',
'10_Portfolios_Prior_12_2_Daily',
'F-F_ST_Reversal_Factor',
'F-F_ST_Reversal_Factor_daily',
'6_Portfolios_ME_Prior_1_0',
'6_Portfolios_ME_Prior_1_0_Daily',
'25_Portfolios_ME_Prior_1_0',
'25_Portfolios_ME_Prior_1_0_Daily',
'10_Portfolios_Prior_1_0',
'10_Portfolios_Prior_1_0_Daily',
'F-F_LT_Reversal_Factor',
```

```
'F-F_LT_Reversal_Factor_daily',
'6_Portfolios_ME_Prior_60_13',
'6_Portfolios_ME_Prior_60_13_Daily',
'25_Portfolios_ME_Prior_60_13',
'25_Portfolios_ME_Prior_60_13_Daily',
'10_Portfolios_Prior_60_13',
'10_Portfolios_Prior_60_13_Daily',
'Portfolios_Formed_on_AC',
'25_Portfolios_ME_AC_5x5',
'Portfolios_Formed_on_BETA',
'25_Portfolios_ME_BETA_5x5',
'Portfolios_Formed_on_NI',
'25_Portfolios_ME_NI_5x5',
'Portfolios_Formed_on_VAR',
'25_Portfolios_ME_VAR_5x5',
'Portfolios_Formed_on_RESVAR',
'25_Portfolios_ME_RESVAR_5x5',
'5_Industry_Portfolios',
'5_Industry_Portfolios_Wout_Div',
'5_Industry_Portfolios_daily',
'10_Industry_Portfolios',
'10_Industry_Portfolios_Wout_Div',
'10_Industry_Portfolios_daily',
'12_Industry_Portfolios',
'12_Industry_Portfolios_Wout_Div',
'12_Industry_Portfolios_daily',
'17_Industry_Portfolios',
'17_Industry_Portfolios_Wout_Div',
'17_Industry_Portfolios_daily',
'30_Industry_Portfolios',
'30_Industry_Portfolios_Wout_Div',
'30_Industry_Portfolios_daily',
'38_Industry_Portfolios',
'38_Industry_Portfolios_Wout_Div',
'38_Industry_Portfolios_daily',
'48_Industry_Portfolios',
'48_Industry_Portfolios_Wout_Div',
'48_Industry_Portfolios_daily',
'49_Industry_Portfolios',
'49_Industry_Portfolios_Wout_Div',
'49_Industry_Portfolios_daily',
'ME_Breakpoints',
'BE-ME_Breakpoints',
'OP_Breakpoints',
'INV_Breakpoints',
'E-P_Breakpoints',
'CF-P_Breakpoints',
```

```
'D-P_Breakpoints',
'Prior_2-12_Breakpoints',
'Developed_3_Factors',
'Developed_3_Factors_Daily',
'Developed_ex_US_3_Factors',
'Developed_ex_US_3_Factors_Daily',
'Europe_3_Factors',
'Europe_3_Factors_Daily',
'Japan_3_Factors',
'Japan_3_Factors_Daily',
'Asia_Pacific_ex_Japan_3_Factors',
'Asia_Pacific_ex_Japan_3_Factors_Daily',
'North_America_3_Factors',
'North_America_3_Factors_Daily',
'Developed_5_Factors',
'Developed_5_Factors_Daily',
'Developed_ex_US_5_Factors',
'Developed_ex_US_5_Factors_Daily',
'Europe_5_Factors',
'Europe_5_Factors_Daily',
'Japan_5_Factors',
'Japan_5_Factors_Daily',
'Asia_Pacific_ex_Japan_5_Factors',
'Asia_Pacific_ex_Japan_5_Factors_Daily',
'North_America_5_Factors',
'North_America_5_Factors_Daily',
'Developed_Mom_Factor',
'Developed_Mom_Factor_Daily',
'Developed_ex_US_Mom_Factor',
'Developed_ex_US_Mom_Factor_Daily',
'Europe_Mom_Factor',
'Europe_Mom_Factor_Daily',
'Japan_Mom_Factor',
'Japan_Mom_Factor_Daily',
'Asia_Pacific_ex_Japan_MOM_Factor',
'Asia_Pacific_ex_Japan_MOM_Factor_Daily',
'North_America_Mom_Factor',
'North_America_Mom_Factor_Daily',
'Developed_6_Portfolios_ME_BE-ME',
'Developed_6_Portfolios_ME_BE-ME_daily',
'Developed_ex_US_6_Portfolios_ME_BE-ME',
'Developed_ex_US_6_Portfolios_ME_BE-ME_daily',
'Europe_6_Portfolios_ME_BE-ME',
'Europe_6_Portfolios_ME_BE-ME_daily',
'Japan_6_Portfolios_ME_BE-ME',
'Japan_6_Portfolios_ME_BE-ME_daily',
'Asia_Pacific_ex_Japan_6_Portfolios_ME_BE-ME',
```

```
'Asia_Pacific_ex_Japan_6_Portfolios_ME_BE-ME_daily',
'North_America_6_Portfolios_ME_BE-ME',
'North_America_6_Portfolios_ME_BE-ME_daily',
'Developed_25_Portfolios_ME_BE-ME',
'Developed_25_Portfolios_ME_BE-ME_daily',
'Developed_ex_US_25_Portfolios_ME_BE-ME',
'Developed_ex_US_25_Portfolios_ME_BE-ME_daily',
'Europe_25_Portfolios_ME_BE-ME',
'Europe_25_Portfolios_ME_BE-ME_daily',
'Japan_25_Portfolios_ME_BE-ME',
'Japan_25_Portfolios_ME_BE-ME_daily',
'Asia_Pacific_ex_Japan_25_Portfolios_ME_BE-ME',
'Asia_Pacific_ex_Japan_25_Portfolios_ME_BE-ME_daily',
'North_America_25_Portfolios_ME_BE-ME',
'North_America_25_Portfolios_ME_BE-ME_daily',
'Developed_6_Portfolios_ME_OP',
'Developed_6_Portfolios_ME_OP_Daily',
'Developed_ex_US_6_Portfolios_ME_OP',
'Developed_ex_US_6_Portfolios_ME_OP_Daily',
'Europe_6_Portfolios_ME_OP',
'Europe_6_Portfolios_ME_OP_Daily',
'Japan_6_Portfolios_ME_OP',
'Japan_6_Portfolios_ME_OP_Daily',
'Asia_Pacific_ex_Japan_6_Portfolios_ME_OP',
'Asia_Pacific_ex_Japan_6_Portfolios_ME_OP_Daily',
'North_America_6_Portfolios_ME_OP',
'North_America_6_Portfolios_ME_OP_Daily',
'Developed_25_Portfolios_ME_OP',
'Developed_25_Portfolios_ME_OP_Daily',
'Developed_ex_US_25_Portfolios_ME_OP',
'Developed_ex_US_25_Portfolios_ME_OP_Daily',
'Europe_25_Portfolios_ME_OP',
'Europe_25_Portfolios_ME_OP_Daily',
'Japan_25_Portfolios_ME_OP',
'Japan_25_Portfolios_ME_OP_Daily',
'Asia_Pacific_ex_Japan_25_Portfolios_ME_OP',
'Asia_Pacific_ex_Japan_25_Portfolios_ME_OP_Daily',
'North_America_25_Portfolios_ME_OP',
'North_America_25_Portfolios_ME_OP_Daily',
'Developed_6_Portfolios_ME_INV',
'Developed_6_Portfolios_ME_INV_Daily',
'Developed_ex_US_6_Portfolios_ME_INV',
'Developed_ex_US_6_Portfolios_ME_INV_Daily',
'Europe_6_Portfolios_ME_INV',
'Europe_6_Portfolios_ME_INV_Daily',
'Japan_6_Portfolios_ME_INV',
'Japan_6_Portfolios_ME_INV_Daily',
```

```
'Asia_Pacific_ex_Japan_6_Portfolios_ME_INV',
'Asia_Pacific_ex_Japan_6_Portfolios_ME_INV_Daily',
'North_America_6_Portfolios_ME_INV',
'North_America_6_Portfolios_ME_INV_Daily',
'Developed_25_Portfolios_ME_INV',
'Developed_25_Portfolios_ME_INV_Daily',
'Developed_ex_US_25_Portfolios_ME_INV',
'Developed_ex_US_25_Portfolios_ME_INV_Daily',
'Europe_25_Portfolios_ME_INV',
'Europe_25_Portfolios_ME_INV_Daily',
'Japan_25_Portfolios_ME_INV',
'Japan_25_Portfolios_ME_INV_Daily',
'Asia_Pacific_ex_Japan_25_Portfolios_ME_INV',
'Asia_Pacific_ex_Japan_25_Portfolios_ME_INV_Daily',
'North_America_25_Portfolios_ME_INV',
'North_America_25_Portfolios_ME_INV_Daily',
'Developed_6_Portfolios_ME_Prior_12_2',
'Developed_6_Portfolios_ME_Prior_250_20_daily',
'Developed_ex_US_6_Portfolios_ME_Prior_12_2',
'Developed_ex_US_6_Portfolios_ME_Prior_250_20_daily',
'Europe_6_Portfolios_ME_Prior_12_2',
'Europe_6_Portfolios_ME_Prior_250_20_daily',
'Japan_6_Portfolios_ME_Prior_12_2',
'Japan_6_Portfolios_ME_Prior_250_20_daily',
'Asia_Pacific_ex_Japan_6_Portfolios_ME_Prior_12_2',
'Asia_Pacific_ex_Japan_6_Portfolios_ME_Prior_250_20_daily',
'North_America_6_Portfolios_ME_Prior_12_2',
'North_America_6_Portfolios_ME_Prior_250_20_daily',
'Developed_25_Portfolios_ME_Prior_12_2',
'Developed_25_Portfolios_ME_Prior_250_20_daily',
'Developed_ex_US_25_Portfolios_ME_Prior_12_2',
'Developed_ex_US_25_Portfolios_ME_Prior_250_20_daily',
'Europe_25_Portfolios_ME_Prior_12_2',
'Europe_25_Portfolios_ME_Prior_250_20_daily',
'Japan_25_Portfolios_ME_Prior_12_2',
'Japan_25_Portfolios_ME_Prior_250_20_daily',
'Asia_Pacific_ex_Japan_25_Portfolios_ME_Prior_12_2',
'Asia_Pacific_ex_Japan_25_Portfolios_ME_Prior_250_20_daily',
'North_America_25_Portfolios_ME_Prior_12_2',
'North_America_25_Portfolios_ME_Prior_250_20_daily',
'Developed_32_Portfolios_ME_BE-ME_OP_2x4x4',
'Developed_ex_US_32_Portfolios_ME_BE-ME_OP_2x4x4',
'Europe_32_Portfolios_ME_BE-ME_OP_2x4x4',
'Japan_32_Portfolios_ME_BE-ME_OP_2x4x4',
'Asia_Pacific_ex_Japan_32_Portfolios_ME_BE-ME_OP_2x4x4',
'North_America_32_Portfolios_ME_BE-ME_OP_2x4x4',
'Developed_32_Portfolios_ME_BE-ME_INV(TA)_2x4x4',
```

```
 'Developed_ex_US_32_Portfolios_ME_BE-ME_INV(TA)_2x4x4',
 'Europe_32_Portfolios_ME_BE-ME_INV(TA)_2x4x4',
 'Japan_32_Portfolios_ME_BE-ME_INV(TA)_2x4x4',
 'Asia_Pacific_ex_Japan_32_Portfolios_ME_BE-ME_INV(TA)_2x4x4',
 'North_America_32_Portfolios_ME_BE-ME_INV(TA)_2x4x4',
 'Developed_32_Portfolios_ME_INV(TA)_OP_2x4x4',
 'Developed_ex_US_32_Portfolios_ME_INV(TA)_OP_2x4x4',
 'Europe_32_Portfolios_ME_INV(TA)_OP_2x4x4',
 'Japan_32_Portfolios_ME_INV(TA)_OP_2x4x4',
 'Asia_Pacific_ex_Japan_32_Portfolios_ME_INV(TA)_OP_2x4x4',
 'North_America_32_Portfolios_ME_INV(TA)_OP_2x4x4',
 'Emerging_5_Factors',
 'Emerging_MOM_Factor',
 'Emerging_Markets_6_Portfolios_ME_BE-ME',
 'Emerging_Markets_6_Portfolios_ME_OP',
 'Emerging_Markets_6_Portfolios_ME_INV',
 'Emerging_Markets_6_Portfolios_ME_Prior_12_2',
 'Emerging_Markets_4_Portfolios_BE-ME_OP',
 'Emerging_Markets_4_Portfolios_OP_INV',
 'Emerging_Markets_4_Portfolios_BE-ME_INV']
```

[21]:
```
ds = web.DataReader('5_Industry_Portfolios', 'famafrench')
print(ds['DESCR'])
```

```
5 Industry Portfolios
---------------------

This file was created by CMPT_IND_RETS using the 202012 CRSP database. It
contains value- and equal-weighted returns for 5 industry portfolios. The
portfolios are constructed at the end of June. The annual returns are from
January to December. Missing data are indicated by -99.99 or -999. Copyright
2020 Kenneth R. French

 0 : Average Value Weighted Returns -- Monthly (59 rows x 5 cols)
 1 : Average Equal Weighted Returns -- Monthly (59 rows x 5 cols)
 2 : Average Value Weighted Returns -- Annual (5 rows x 5 cols)
 3 : Average Equal Weighted Returns -- Annual (5 rows x 5 cols)
 4 : Number of Firms in Portfolios (59 rows x 5 cols)
 5 : Average Firm Size (59 rows x 5 cols)
 6 : Sum of BE / Sum of ME (5 rows x 5 cols)
 7 : Value-Weighted Average of BE/ME (5 rows x 5 cols)
```

1.3.6 World Bank

[22]:
```
from pandas_datareader import wb
gdp_variables = wb.search('gdp.*capita.*const')
gdp_variables.head()
```

[22]:
```
                        id                                                name  \
680     6.0.GDPpc_constant  GDP per capita, PPP (constant 2011 internation…
9266        NY.GDP.PCAP.KD                 GDP per capita (constant 2010 US$)
9268        NY.GDP.PCAP.KN                      GDP per capita (constant LCU)
9270     NY.GDP.PCAP.PP.KD  GDP per capita, PPP (constant 2017 internation…
9271  NY.GDP.PCAP.PP.KD.87  GDP per capita, PPP (constant 1987 internation…

     unit                          source  \
680                       LAC Equity Lab
9266       World Development Indicators
9268       World Development Indicators
9270       World Development Indicators
9271              WDI Database Archives

                                              sourceNote  \
680   GDP per capita based on purchasing power parit…
9266  GDP per capita is gross domestic product divid…
9268  GDP per capita is gross domestic product divid…
9270  GDP per capita based on purchasing power parit…
9271

                                    sourceOrganization            topics
680        b'World Development Indicators (World Bank)'  Economy & Growth
9266  b'World Bank national accounts data, and OECD …  Economy & Growth
9268  b'World Bank national accounts data, and OECD …  Economy & Growth
9270  b'International Comparison Program, World Bank…  Economy & Growth
9271                                              b''
```

[23]:
```
wb_data = wb.download(indicator='NY.GDP.PCAP.KD',
                      country=['US', 'CA', 'MX'],
                      start=1990,
                      end=2019)
wb_data.head()
```

[23]:
```
              NY.GDP.PCAP.KD
country year
Canada  2019    51588.761434
        2018    51476.200774
        2017    51170.475841
        2016    50193.750410
        2015    50262.027666
```

1.3.7 OECD

[24]:
```
df = web.DataReader('TUD', 'oecd', start='2010', end='2019')
df[['Japan', 'United States']]
```

[24]:
```
Country                 Japan                                                \
Source     Administrative data                                    Survey data
Series               Employees Union members Trade union  density   Employees
Year
2010-01-01             37100.0       12509.0                 33.7         NaN
2011-01-01             37460.0       12437.0                 33.2         NaN
2012-01-01             38990.0       12309.0                 31.6         NaN
2013-01-01             40120.0       12369.0                 30.8         NaN
2014-01-01             41020.0       12526.0                 30.5         NaN
2015-01-01             42090.0       12520.0                 29.7         NaN
2016-01-01             43010.0       12418.0                 28.9         NaN
2017-01-01             43830.0       12343.0                 28.2         NaN
2018-01-01             45650.0       12227.0                 26.8         NaN

Country                                          United States  \
Source                                        Administrative data
Series     Union members Trade union  density           Employees
Year
2010-01-01           NaN                 NaN             80520.0
2011-01-01           NaN                 NaN             83482.0
2012-01-01           NaN                 NaN             89675.0
2013-01-01           NaN                 NaN             89950.0
2014-01-01           NaN                 NaN                 NaN
2015-01-01           NaN                 NaN                 NaN
2016-01-01           NaN                 NaN                 NaN
2017-01-01           NaN                 NaN                 NaN
2018-01-01           NaN                 NaN                 NaN

Country                                                                \
Source                                        Survey data
Series     Union members Trade union  density   Employees Union members
Year
2010-01-01       19634.0                 24.4     80519.0       17403.0
2011-01-01       19695.0                 23.6     83481.0       19335.0
2012-01-01       20055.0                 22.4     89673.0       20986.0
2013-01-01       19843.0                 22.1     89950.0       20095.0
2014-01-01           NaN                 NaN         NaN           NaN
2015-01-01           NaN                 NaN     91079.0       17717.0
2016-01-01           NaN                 NaN     97406.0       16996.0
2017-01-01           NaN                 NaN     99846.0       16975.0
2018-01-01           NaN                 NaN    104642.0       17002.0
```

```
Country
Source
Series     Trade union  density
Year
2010-01-01                 21.6
2011-01-01                 23.2
2012-01-01                 23.4
2013-01-01                 22.3
2014-01-01                  NaN
2015-01-01                 19.5
2016-01-01                 17.4
2017-01-01                 17.0
2018-01-01                 16.2
```

1.3.8 Stooq

Google finance stopped providing common index data download. The Stooq site had this data for download for a while but is currently broken, awaiting release of fix

[25]:
```
index_url = 'https://stooq.com/t/'
ix = pd.read_html(index_url)
len(ix)
```

[25]: 47

[26]:
```
sp500_stooq = web.DataReader('^SPX', 'stooq')
sp500_stooq.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1258 entries, 2021-02-22 to 2016-02-24
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Open    1258 non-null   float64
 1   High    1258 non-null   float64
 2   Low     1258 non-null   float64
 3   Close   1258 non-null   float64
 4   Volume  1258 non-null   int64
dtypes: float64(4), int64(1)
memory usage: 59.0 KB
```

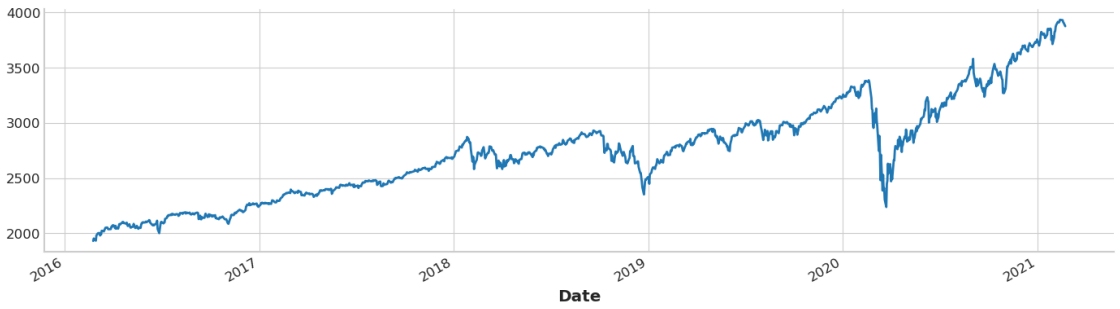
[27]:
```
sp500_stooq.head()
```

[27]:

Date	Open	High	Low	Close	Volume
2021-02-22	3885.55	3902.92	3874.71	3876.50	2748914392
2021-02-19	3921.16	3930.41	3903.07	3906.71	2315685076

```
2021-02-18  3915.86  3921.98  3885.03  3913.97  2025989354
2021-02-17  3918.50  3933.61  3900.43  3931.33  2161952392
2021-02-16  3939.61  3950.43  3923.85  3932.59  2305147298
```

[28]:
```
sp500_stooq.Close.plot(figsize=(14,4))
sns.despine()
plt.tight_layout()
```



1.3.9 NASDAQ Symbols

[29]:
```
from pandas_datareader.nasdaq_trader import get_nasdaq_symbols
symbols = get_nasdaq_symbols()
symbols.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9897 entries, A to ZYXI
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Nasdaq Traded     9897 non-null   bool
 1   Security Name     9897 non-null   object
 2   Listing Exchange  9897 non-null   category
 3   Market Category   9897 non-null   object
 4   ETF               9897 non-null   bool
 5   Round Lot Size    9897 non-null   float64
 6   Test Issue        9897 non-null   bool
 7   Financial Status  4191 non-null   category
 8   CQS Symbol        5706 non-null   object
 9   NASDAQ Symbol     9897 non-null   object
 10  NextShares        9897 non-null   bool
dtypes: bool(4), category(2), float64(1), object(4)
memory usage: 522.2+ KB
```

1.3.10 Tiingo

Requires signing up and storing API key in environment

[30]:
```
df = web.get_data_tiingo('GOOG', api_key=os.getenv('TIINGO_API_KEY'))
```

[31]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 1258 entries, ('GOOG', Timestamp('2016-02-24 00:00:00+0000',
tz='UTC')) to ('GOOG', Timestamp('2021-02-22 00:00:00+0000', tz='UTC'))
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   close        1258 non-null   float64
 1   high         1258 non-null   float64
 2   low          1258 non-null   float64
 3   open         1258 non-null   float64
 4   volume       1258 non-null   int64
 5   adjClose     1258 non-null   float64
 6   adjHigh      1258 non-null   float64
 7   adjLow       1258 non-null   float64
 8   adjOpen      1258 non-null   float64
 9   adjVolume    1258 non-null   int64
 10  divCash      1258 non-null   float64
 11  splitFactor  1258 non-null   float64
dtypes: float64(10), int64(2)
memory usage: 164.0+ KB
```