pandas-datareader Documentation

Release 0.10.0

pydata

Jul 13, 2021

Contents

Version: **0.10.0** Date: **July 13, 2021**

Up-to-date remote data access for pandas. Works for multiple versions of pandas.

CHAPTER 1

Quick Start

Install using `pip`

```
pip install pandas-datareader
```

and then import and use one of the data readers. This example reads 5-years of 10-year constant maturity yields on U.S. government bonds.

```
import pandas_datareader as pdr
pdr.get_data_fred('GS10')
```

CHAPTER 2

Contents

Contents:

2.1 Remote Data Access

Warning: The `access_key` keyword argument of `DataReader` has been deprecated in favor of `api_key`.

Functions from `pandas_datareader.data` and *`pandas_datareader.wb`* extract data from various Internet sources into a pandas DataFrame. Currently the following sources are supported:

- *Tiingo*
- *IEX*
- *Alpha Vantage*
- *Econdb*
- *Enigma*
- *Quandl*
- *St.Louis FED (FRED)*
- *Kenneth French's data library*
- *World Bank*
- *OECD*
- *Eurostat*
- *Thrift Savings Plan*
- *Nasdaq Trader symbol definitions*
- *Stooq*

- *MOEX*
- *Naver Finance*
- *Yahoo Finance*

It should be noted, that various sources support different kinds of data, so not all sources implement the same methods and the data elements returned might also differ.

2.1.1 Tiingo

Tiingo is a tracing platform that provides a data api with historical end-of-day prices on equities, mutual funds and ETFs. Free registration is required to get an API key. Free accounts are rate limited and can access a limited number of symbols (500 at the time of writing).

```
In [1]: import os
In [2]: import pandas_datareader as pdr

In [3]: df = pdr.get_data_tiingo('GOOG', api_key=os.getenv('TIINGO_API_KEY'))
In [4]: df.head()
                                close    high     low     open  volume  adjClose␣
↪adjHigh  adjLow  adjOpen  adjVolume  divCash  splitFactor
symbol date
GOOG   2014-03-27 00:00:00+00:00  558.46  568.00  552.92  568.000   13100    558.46␣
↪568.00  552.92  568.000      13100      0.0          1.0
       2014-03-28 00:00:00+00:00  559.99  566.43  558.67  561.200   41100    559.99␣
↪566.43  558.67  561.200      41100      0.0          1.0
       2014-03-31 00:00:00+00:00  556.97  567.00  556.93  566.890   10800    556.97␣
↪567.00  556.93  566.890      10800      0.0          1.0
       2014-04-01 00:00:00+00:00  567.16  568.45  558.71  558.710    7900    567.16␣
↪568.45  558.71  558.710       7900      0.0          1.0
       2014-04-02 00:00:00+00:00  567.00  604.83  562.19  565.106  146700    567.00␣
↪604.83  562.19  565.106     146700      0.0          1.0
```

2.1.2 IEX

Warning: Usage of all IEX readers now requires an API key. See below for additional information.

The Investors Exchange (IEX) provides a wide range of data through an API. Historical stock prices are available for up to 15 years. The usage of these readers requires the publishable API key from IEX Cloud Console, which can be stored in the `IEX_API_KEY` environment variable.

```
In [1]: import pandas_datareader.data as web

In [2]: from datetime import datetime

In [3]: start = datetime(2016, 9, 1)

In [4]: end = datetime(2018, 9, 1)

In [5]: f = web.DataReader('F', 'iex', start, end)

In [6]: f.loc['2018-08-31']
```

(continues on next page)

(continued from previous page)

```
Out[6]:
open              9.64
high              9.68
low               9.40
close             9.48
volume    76424884.00
Name: 2018-08-31, dtype: float64
```

Note: You must provide an API Key when using IEX. You can do this using `os.environ["IEX_API_KEY"] = "pk_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"` or by exporting the key before starting the IPython session.

There are additional interfaces to this API that are directly exposed: tops (*'iex-tops'*) and last (*'iex-lasts'*). A third interface to the deep API is exposed through *Deep* class or the *get_iex_book* function.

Todo: Execute block when markets are open

```
import pandas_datareader.data as web
f = web.DataReader('gs', 'iex-tops')
f[:10]
```

2.1.3 Alpha Vantage

Alpha Vantage provides realtime equities and forex data. Free registration is required to get an API key.

Historical Time Series Data

Through the Alpha Vantage Time Series endpoints, it is possible to obtain historical equities and currency rate data for individual symbols. For daily, weekly, and monthly frequencies, 20+ years of historical data is available. The past 3-5 days of intraday data is also available.

The following endpoints are available:

- `av-intraday` - Intraday Time Series
- `av-daily` - Daily Time Series
- `av-daily-adjusted` - Daily Time Series (Adjusted)
- `av-weekly` - Weekly Time Series
- `av-weekly-adjusted` - Weekly Time Series (Adjusted)
- `av-monthly` - Monthly Time Series
- `av-monthly-adjusted` - Monthly Time Series (Adjusted)
- `av-forex-daily` - Daily Time Series

```
In [1]: import os

In [2]: from datetime import datetime

In [3]: import pandas_datareader.data as web
```

(continues on next page)

(continued from previous page)

```
In [4]: f = web.DataReader("AAPL", "av-daily", start=datetime(2017, 2, 9),
   ...:                    end=datetime(2017, 5, 24),
   ...:                    api_key=os.getenv('ALPHAVANTAGE_API_KEY'))

In [5]: f.loc["2017-02-09"]
Out[5]:
open      1.316500e+02
high      1.324450e+02
low       1.311200e+02
close     1.324200e+02
volume    2.834990e+07
Name: 2017-02-09, dtype: float64
```

To request the historical exchange rate of physical or digital currencies, use `av-forex-daily` and format the symbol as "FROM/TO", for example "USD/JPY".

The top-level function `get_data_alphavantage` is also provided. This function will return the `TIME_SERIES_DAILY` endpoint for the symbol and date range provided.

Quotes

Alpha Vantage Batch Stock Quotes endpoint allows the retrieval of realtime stock quotes for up to 100 symbols at once. These quotes are accessible through the top-level function `get_quote_av`.

```
In [1]: import os

In [2]: from datetime import datetime

In [3]: import pandas_datareader.data as web

In [4]: web.get_quote_av(["AAPL", "TSLA"])
Out[4]:
         price  volume            timestamp
symbol
AAPL    219.87     NaN  2019-09-16 15:59:59
TSLA    242.80     NaN  2019-09-16 15:59:57
```

Note: Most quotes are only available during market hours.

Forex

Alpha Vantage provides realtime currency exchange rates (for physical and digital currencies).

To request the exchange rate of physical or digital currencies, simply format as "FROM/TO" as in "USD/JPY".

```
In [1]: import os

In [2]: import pandas_datareader.data as web

In [3]: f = web.DataReader("USD/JPY", "av-forex",
    ...:                   api_key=os.getenv('ALPHAVANTAGE_API_KEY'))
```

(continues on next page)

(continued from previous page)

```
In [4]: f
Out[4]:
                                USD/JPY
From_Currency Code                  USD
From_Currency Name  United States Dollar
To_Currency Code                    JPY
To_Currency Name            Japanese Yen
Exchange Rate                108.17000000
Last Refreshed        2019-09-17 10:43:36
Time Zone                           UTC
Bid Price                    108.17000000
Ask Price                    108.17000000
```

Multiple pairs are are allowable:

```
In [1]: import os

In [2]: import pandas_datareader.data as web

In [3]: f = web.DataReader(["USD/JPY", "BTC/CNY"], "av-forex",
   ...:                    api_key=os.getenv('ALPHAVANTAGE_API_KEY'))


In [4]: f
Out[4]:
                                USD/JPY              BTC/CNY
From_Currency Code                  USD                  BTC
From_Currency Name  United States Dollar              Bitcoin
To_Currency Code                    JPY                  CNY
To_Currency Name            Japanese Yen         Chinese Yuan
Exchange Rate                108.17000000       72230.38039500
Last Refreshed        2019-09-17 10:44:35  2019-09-17 10:44:01
Time Zone                           UTC                  UTC
Bid Price                    108.17000000       72226.26407700
Ask Price                    108.17000000       72230.02554000
```

Sector Performance

Alpha Vantage provides sector performances through the top-level function `get_sector_performance_av`.

```
In [1]: import os

In [2]: import pandas_datareader.data as web

In [3]: web.get_sector_performance_av().head()
Out[4]:
                  RT      1D      5D      1M     3M      YTD       1Y      3Y       5Y ␣
↪    10Y
Energy         3.29%   3.29%   4.82%  11.69%  3.37%   9.07%  -15.26%  -7.69%  -32.31% ␣
↪ 12.15%
Real Estate    1.02%   1.02%  -1.39%   1.26%  3.49%  24.95%   16.55%     NaN      NaN ␣
↪    NaN
Utilities      0.08%   0.08%   0.72%   2.77%  3.72%  18.16%   16.09%  27.95%   48.41% ␣
↪113.09%
Industrials   -0.15%  -0.15%   2.42%   8.59%  5.10%  22.70%    0.50%  34.50%   43.53% ␣
↪183.47%
```

(continues on next page)

(continued from previous page)

```
Health Care  -0.23%  -0.23%   0.88%   1.91%  0.09%   5.20%   -2.38%  26.37%   43.43%
↪216.01%
```

2.1.4 Econdb

Econdb provides economic data from 90+ official statistical agencies. Free API allows access to the complete Econdb database of time series aggregated into datasets.

Reading data for a single series such as the RGDP for United States, is as simple as taking the ticker segment from the URL path (`RGDPUS` in https://www.econdb.com/series/RGDPUS/) and passing it in like:

```
import os
import pandas_datareader.data as web

f = web.DataReader('ticker=RGDPUS', 'econdb')
f.head()
```

The code snippet for exporting the whole dataset, or its filtered down subset, can be generated by using the Export -> Pandas Python3 functionality on any of the numerous datasets available, such as the Eurostat's GDP and main components

```
import os
import pandas_datareader.data as web

df = web.DataReader('dataset=NAMQ_10_GDP&v=Geopolitical entity (reporting)&h=TIME&
↪from=2018-05-01&to=2021-01-01&GEO=[AL,AT,BE,BA,BG,HR,CY,CZ,DK,EE,EA19,FI,FR,DE,EL,
↪HU,IS,IE,IT,XK,LV,LT,LU,MT,ME,NL,MK,NO,PL,PT,RO,RS,SK,SI,ES,SE,CH,TR,UK]&NA_
↪ITEM=[B1GQ]&S_ADJ=[SCA]&UNIT=[CLV10_MNAC]', 'econdb')
df.columns
```

Datasets can be located through Econdb's search engine, or discovered by exploring the tree of available statistical sources.

2.1.5 Enigma

Access datasets from Enigma, the world's largest repository of structured public data. Note that the Enigma URL has changed from app.enigma.io as of release `0.6.0`, as the old API deprecated.

Datasets are unique identified by the `uuid4` at the end of a dataset's web address. For example, the following code downloads from USDA Food Recalls 1996 Data.

```
In [1]: import os

In [2]: import pandas_datareader as pdr

In [3]: df = pdr.get_data_enigma('292129b0-1275-44c8-a6a3-2a0881f24fe1', os.getenv(
↪'ENIGMA_API_KEY'))

In [4]: df.columns
Out[4]:
Index(['case_number', 'recall_notification_report_number',
       'recall_notification_report_url', 'date_opened', 'date_closed',
       'recall_class', 'press_release', 'domestic_est_number', 'company_name',
       'imported_product', 'foreign_estab_number', 'city', 'state', 'country',
```

(continues on next page)

(continued from previous page)

```
       'product', 'problem', 'description', 'total_pounds_recalled',
       'pounds_recovered'],
      dtype='object')
```

2.1.6 Quandl

Daily financial data (prices of stocks, ETFs etc.) from Quandl. The symbol names consist of two parts: DB name and symbol name. DB names can be all the free ones listed on the Quandl website. Symbol names vary with DB name; for WIKI (US stocks), they are the common ticker symbols, in some other cases (such as FSE) they can be a bit strange. Some sources are also mapped to suitable ISO country codes in the dot suffix style shown above, currently available for BE, CN, DE, FR, IN, JP, NL, PT, UK, US.

As of June 2017, each DB has a different data schema, the coverage in terms of time range is sometimes surprisingly small, and the data quality is not always good.

```
In [1]: import pandas_datareader.data as web

In [2]: symbol = 'WIKI/AAPL'  # or 'AAPL.US'

In [3]: df = web.DataReader(symbol, 'quandl', '2015-01-01', '2015-01-05')

In [4]: df.loc['2015-01-02']
Out[4]:
              Open    High     Low   Close      Volume  ...     AdjOpen     AdjHigh ␣
↪   AdjLow    AdjClose   AdjVolume
Date                                                    ...
2015-01-02  111.39  111.44  107.35  109.33  53204626.0  ...  105.820966  105.868466 ␣
↪101.982949  103.863957  53204626.0
```

2.1.7 FRED

```
In [1]: import pandas_datareader.data as web

In [2]: import datetime

In [3]: start = datetime.datetime(2010, 1, 1)

In [4]: end = datetime.datetime(2013, 1, 27)

In [5]: gdp = web.DataReader('GDP', 'fred', start, end)

In [6]: gdp.loc['2013-01-01']
Out[6]:
GDP    16569.591
Name: 2013-01-01 00:00:00, dtype: float64

# Multiple series:
In [7]: inflation = web.DataReader(['CPIAUCSL', 'CPILFESL'], 'fred', start, end)

In [8]: inflation.head()
Out[8]:
            CPIAUCSL  CPILFESL
DATE
```

(continues on next page)

(continued from previous page)

```
2010-01-01   217.488   220.633
2010-02-01   217.281   220.731
2010-03-01   217.353   220.783
2010-04-01   217.403   220.822
2010-05-01   217.290   220.962
```

2.1.8 Fama/French

Access datasets from the Fama/French Data Library. The `get_available_datasets` function returns a list of all available datasets.

```
In [9]: from pandas_datareader.famafrench import get_available_datasets

In [10]: import pandas_datareader.data as web

In [11]: len(get_available_datasets())
Out[11]: 297

In [12]: ds = web.DataReader('5_Industry_Portfolios', 'famafrench')

In [13]: print(ds['DESCR'])
5 Industry Portfolios
---------------------

This file was created by CMPT_IND_RETS using the 202105 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 2021 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)

In [14]: ds[4].head()
Out[14]:
         Cnsmr  Manuf  HiTec  Hlth   Other
Date
2016-07    539    622    719    620   1109
2016-08    536    621    713    614   1099
2016-09    534    615    705    609   1090
2016-10    530    613    699    604   1087
2016-11    529    611    688    600   1084
```

2.1.9 World Bank

`pandas` users can easily access thousands of panel data series from the World Bank's World Development Indicators by using the `wb` I/O functions.

Indicators

Either from exploring the World Bank site, or using the search function included, every world bank indicator is accessible.

For example, if you wanted to compare the Gross Domestic Products per capita in constant dollars in North America, you would use the `search` function:

```
In [1]: from pandas_datareader import wb
In [2]: matches = wb.search('gdp.*capita.*const')
```

Then you would use the `download` function to acquire the data from the World Bank's servers:

```
In [3]: dat = wb.download(indicator='NY.GDP.PCAP.KD', country=['US', 'CA', 'MX'],␣
↪start=2005, end=2008)

In [4]: print(dat)
                     NY.GDP.PCAP.KD
country       year
Canada        2008  36005.5004978584
              2007  36182.9138439757
              2006  35785.9698172849
              2005  35087.8925933298
Mexico        2008  8113.10219480083
              2007  8119.21298908649
              2006  7961.96818458178
              2005  7666.69796097264
United States 2008  43069.5819857208
              2007  43635.5852068142
              2006   43228.111147107
              2005  42516.3934699993
```

The resulting dataset is a properly formatted `DataFrame` with a hierarchical index, so it is easy to apply `.groupby` transformations to it:

```
In [6]: dat['NY.GDP.PCAP.KD'].groupby(level=0).mean()
Out[6]:
country
Canada           35765.569188
Mexico            7965.245332
United States    43112.417952
dtype: float64
```

Now imagine you want to compare GDP to the share of people with cellphone contracts around the world.

```
In [7]: wb.search('cell.*%').iloc[:,:2]
Out[7]:
                    id                                                name
3990  IT.CEL.SETS.FE.ZS  Mobile cellular telephone users, female (% of ...
3991  IT.CEL.SETS.MA.ZS  Mobile cellular telephone users, male (% of po...
4027      IT.MOB.COV.ZS  Population coverage of mobile cellular telepho...
```

Notice that this second search was much faster than the first one because `pandas` now has a cached list of available data series.

```
In [13]: ind = ['NY.GDP.PCAP.KD', 'IT.MOB.COV.ZS']
In [14]: dat = wb.download(indicator=ind, country='all', start=2011, end=2011).
↪dropna()
```

(continues on next page)

(continued from previous page)

```
In [15]: dat.columns = ['gdp', 'cellphone']
In [16]: print(dat.tail())
                        gdp  cellphone
country   year
Swaziland 2011  2413.952853       94.9
Tunisia   2011  3687.340170      100.0
Uganda    2011   405.332501      100.0
Zambia    2011   767.911290       62.0
Zimbabwe  2011   419.236086       72.4
```

Finally, we use the `statsmodels` package to assess the relationship between our two variables using ordinary least squares regression. Unsurprisingly, populations in rich countries tend to use cellphones at a higher rate:

```
In [17]: import numpy as np
In [18]: import statsmodels.formula.api as smf
In [19]: mod = smf.ols('cellphone ~ np.log(gdp)', dat).fit()
In [20]: print(mod.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:              cellphone   R-squared:                       0.297
Model:                            OLS   Adj. R-squared:                  0.274
Method:                 Least Squares   F-statistic:                     13.08
Date:                Thu, 25 Jul 2013   Prob (F-statistic):            0.00105
Time:                        15:24:42   Log-Likelihood:                -139.16
No. Observations:                  33   AIC:                             282.3
Df Residuals:                      31   BIC:                             285.3
Df Model:                           1
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
Intercept     16.5110     19.071      0.866      0.393       -22.384    55.406
np.log(gdp)    9.9333      2.747      3.616      0.001         4.331    15.535
==============================================================================
Omnibus:                       36.054   Durbin-Watson:                   2.071
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              119.133
Skew:                          -2.314   Prob(JB):                     1.35e-26
Kurtosis:                      11.077   Cond. No.                         45.8
==============================================================================
```

Country Codes

The `country` argument accepts a string or list of mixed two or three character ISO country codes, as well as dynamic World Bank exceptions to the ISO standards.

For a list of the the hard-coded country codes (used solely for error handling logic) see `pandas_datareader.wb.country_codes`.

Problematic Country Codes & Indicators

Note: The World Bank's country list and indicators are dynamic. As of 0.15.1, *`wb.download()`* is more flexible. To achieve this, the warning and exception logic changed.

The world bank converts some country codes, in their response, which makes error checking by pandas difficult. Retired indicators still persist in the search.

Given the new flexibility of 0.15.1, improved error handling by the user may be necessary for fringe cases.

To help identify issues:

There are at least 4 kinds of country codes:

1. Standard (2/3 digit ISO) - returns data, will warn and error properly.
2. Non-standard (WB Exceptions) - returns data, but will falsely warn.
3. Blank - silently missing from the response.
4. Bad - causes the entire response from WB to fail, always exception inducing.

There are at least 3 kinds of indicators:

1. Current - Returns data.
2. Retired - Appears in search results, yet won't return data.
3. Bad - Will not return data.

Use the `errors` argument to control warnings and exceptions. Setting errors to ignore or warn, won't stop failed responses. (ie, 100% bad indicators, or a single 'bad' (#4 above) country code).

See docstrings for more info.

2.1.10 OECD

OECD Statistics are available via `DataReader`. You have to specify OECD's data set code.

To confirm data set code, access to `each data -> Export -> SDMX Query`. Following example is to download 'Trade Union Density' data which set code is 'TUD'.

```
In [15]: import pandas_datareader.data as web

In [16]: import datetime

In [17]: df = web.DataReader('TUD', 'oecd')

In [18]: df.columns
Out[18]:
MultiIndex([(      'Australia', 'Annual', 'Percentage of employees'),
            (        'Austria', 'Annual', 'Percentage of employees'),
            (        'Belgium', 'Annual', 'Percentage of employees'),
            (         'Canada', 'Annual', 'Percentage of employees'),
            ( 'Czech Republic', 'Annual', 'Percentage of employees'),
            (        'Denmark', 'Annual', 'Percentage of employees'),
            (        'Finland', 'Annual', 'Percentage of employees'),
            (         'France', 'Annual', 'Percentage of employees'),
            (        'Germany', 'Annual', 'Percentage of employees'),
            (         'Greece', 'Annual', 'Percentage of employees'),
            (        'Hungary', 'Annual', 'Percentage of employees'),
            (        'Iceland', 'Annual', 'Percentage of employees'),
            (        'Ireland', 'Annual', 'Percentage of employees'),
            (          'Italy', 'Annual', 'Percentage of employees'),
            (          'Japan', 'Annual', 'Percentage of employees'),
            (          'Korea', 'Annual', 'Percentage of employees'),
```

(continues on next page)

(continued from previous page)

```
            (     'Luxembourg', 'Annual', 'Percentage of employees'),
            (         'Mexico', 'Annual', 'Percentage of employees'),
            (    'Netherlands', 'Annual', 'Percentage of employees'),
            (    'New Zealand', 'Annual', 'Percentage of employees'),
            (         'Norway', 'Annual', 'Percentage of employees'),
            (         'Poland', 'Annual', 'Percentage of employees'),
            (       'Portugal', 'Annual', 'Percentage of employees'),
            ('Slovak Republic', 'Annual', 'Percentage of employees'),
            (          'Spain', 'Annual', 'Percentage of employees'),
            (         'Sweden', 'Annual', 'Percentage of employees'),
            (    'Switzerland', 'Annual', 'Percentage of employees'),
            (         'Turkey', 'Annual', 'Percentage of employees'),
            ( 'United Kingdom', 'Annual', 'Percentage of employees'),
            (  'United States', 'Annual', 'Percentage of employees'),
            (   'OECD - Total', 'Annual', 'Percentage of employees'),
            (          'Chile', 'Annual', 'Percentage of employees'),
            (       'Colombia', 'Annual', 'Percentage of employees'),
            (     'Costa Rica', 'Annual', 'Percentage of employees'),
            (        'Estonia', 'Annual', 'Percentage of employees'),
            (         'Israel', 'Annual', 'Percentage of employees'),
            (         'Latvia', 'Annual', 'Percentage of employees'),
            (      'Lithuania', 'Annual', 'Percentage of employees'),
            (       'Slovenia', 'Annual', 'Percentage of employees')],
           names=['Country', 'Frequency', 'Measure'])

In [19]: df[['Japan', 'United States']]
Out[19]:
Country                      Japan           United States
Frequency                   Annual                  Annual
Measure    Percentage of employees Percentage of employees
Time
2017-01-01               17.500000                    10.6
2018-01-01               17.200001                    10.3
2019-01-01               16.799999                     9.9
2020-01-01                     NaN                    10.3
```

2.1.11 Eurostat

Eurostat are available via `DataReader`.

Get Rail accidents by type of accident (ERA data) data. The result will be a `DataFrame` which has `DatetimeIndex` as index and `MultiIndex` of attributes or countries as column. The target URL is:

- http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=tran_sf_railac&lang=en

You can specify dataset ID 'tran_sf_railac' to get corresponding data via `DataReader`.

```
In [20]: import pandas_datareader.data as web

In [21]: df = web.DataReader('tran_sf_railac', 'eurostat')

In [22]: df
Out[22]:
UNIT                                                                    ␣
↪    Number  ...
ACCIDENT    Collisions of trains, including collisions with obstacles within the␣
↪clearance gauge  ...        Unknown
```

(continues on next page)

(continued from previous page)

```
GEO                                                                                              ␣
↪   Austria  ... United Kingdom
FREQ                                                                                             ␣
↪    Annual  ...         Annual
TIME_PERIOD                                                                                      ␣
↪            ...
2017-01-01                                                        7.0                            ␣
↪            ...            NaN
2018-01-01                                                        4.0                            ␣
↪            ...            NaN
2019-01-01                                                        1.0                            ␣
↪            ...            NaN

[3 rows x 264 columns]
```

2.1.12 Thrift Savings Plan (TSP) Fund Data

Download mutual fund index prices for the Thrift Savings Plan (TSP).

```
In [23]: import pandas_datareader.tsp as tsp

In [24]: tspreader = tsp.TSPReader(start='2015-10-1', end='2015-12-31')

In [25]: tspreader.read()
Out[25]:
            L Income  L 2025   L 2030  ...   C Fund   S Fund   I Fund
Date                                   ...
2015-12-31   17.7733     NaN  25.0635  ...  27.5622  35.2356  24.0952
2015-12-30   17.8066     NaN  25.2267  ...  27.8239  35.5126  24.4184
2015-12-29   17.8270     NaN  25.3226  ...  28.0236  35.8047  24.4757
2015-12-28   17.7950     NaN  25.1691  ...  27.7230  35.4625  24.2816
2015-12-24   17.7991     NaN  25.2052  ...  27.7831  35.6084  24.3272
...              ...     ...      ...  ...      ...      ...      ...
2015-10-07   17.6639     NaN  24.8629  ...  26.7751  35.6035  24.3671
2015-10-06   17.6338     NaN  24.7268  ...  26.5513  35.1320  24.2294
2015-10-05   17.6395     NaN  24.7571  ...  26.6467  35.3565  24.1475
2015-10-02   17.5707     NaN  24.4472  ...  26.1669  34.6504  23.6367
2015-10-01   17.5164     NaN  24.2159  ...  25.7953  34.0993  23.3202

[62 rows x 15 columns]
```

2.1.13 Nasdaq Trader Symbol Definitions

Download the latest symbols from Nasdaq.

Note that Nasdaq updates this file daily, and historical versions are not available. More information on the field definitions.

```
In [12]: from pandas_datareader.nasdaq_trader import get_nasdaq_symbols
In [13]: symbols = get_nasdaq_symbols()
In [14]: print(symbols.loc['IBM'])
    Nasdaq Traded                                                    True
    Security Name       International Business Machines Corporation Co...
```

(continues on next page)

(continued from previous page)

```
    Listing Exchange                                                    N
    Market Category
    ETF                                                             False
    Round Lot Size                                                    100
    Test Issue                                                      False
    Financial Status                                                  NaN
    CQS Symbol                                                        IBM
    NASDAQ Symbol                                                     IBM
    NextShares                                                      False
    Name: IBM, dtype: object
```

2.1.14 Stooq Index Data

Google finance doesn't provide common index data download. The Stooq site has the data for download.

```
In [26]: import pandas_datareader.data as web

In [27]: f = web.DataReader('^DJI', 'stooq')

In [28]: f[:10]
Out[28]:
                Open      High       Low     Close     Volume
Date
2021-07-12  34836.75  35014.90  34730.15  34996.18  344606907
2021-07-09  34457.51  34893.72  34457.51  34870.16  340542786
2021-07-08  34569.01  34569.01  34145.59  34421.93  374853414
2021-07-07  34604.17  34708.78  34435.59  34681.79  340215866
2021-07-06  34790.16  34814.20  34358.42  34577.37  390545107
2021-07-02  34642.42  34821.93  34613.49  34786.35  332517041
2021-07-01  34507.32  34640.28  34498.85  34633.53  309690314
2021-06-30  34290.74  34553.16  34245.48  34502.51  333493947
2021-06-29  34338.89  34469.83  34266.83  34292.29  321388212
2021-06-28  34428.10  34449.65  34186.13  34283.27  320257590
```

2.1.15 MOEX Data

The Moscow Exchange (MOEX) provides historical data.

`pandas_datareader.get_data_moex(*args)` is equivalent to `pandas_datareader.moex.MoexReader(*args).read()`

```
In [29]: import pandas_datareader as pdr

In [30]: f = pdr.get_data_moex(['USD000UTSTOM', 'MAGN'], '2020-07-02', '2020-07-07')

In [31]: f.head()
Out[31]:
            ADMITTEDQUOTE  ADMITTEDVALUE  ...  YIELDLASTCOUPON  YIELDTOOFFER
TRADEDATE                                 ...
2020-07-02            NaN            NaN  ...              NaN           NaN
2020-07-03            NaN            NaN  ...              NaN           NaN
2020-07-06            NaN            NaN  ...              NaN           NaN
2020-07-07            NaN            NaN  ...              NaN           NaN
```

(continues on next page)

(continued from previous page)

```
2020-07-02         37.605    670695507.0  ...              NaN           NaN

[5 rows x 66 columns]

In [32]: f = pdr.moex.MoexReader('SBER', '2020-07-02', '2020-07-03').read()

In [33]: f.head()
Out[33]:
            ADMITTEDQUOTE  ADMITTEDVALUE  ...  YIELDLASTCOUPON  YIELDTOOFFER
TRADEDATE                                 ...
2020-07-02         210.00   1.514438e+10  ...              NaN           NaN
2020-07-03         210.08   1.036950e+10  ...              NaN           NaN

[2 rows x 65 columns]

In [34]: f = pdr.moex.MoexReader('SBER', '2020-07-02', '2020-07-03').read_all_boards()

In [35]: f.head()
Out[35]:
            ADMITTEDQUOTE  ADMITTEDVALUE  ...  YIELDLASTCOUPON  YIELDTOOFFER
TRADEDATE                                 ...
2020-07-02            NaN            NaN  ...              NaN           NaN
2020-07-03            NaN            NaN  ...              NaN           NaN
2020-07-02            NaN            NaN  ...              NaN           NaN
2020-07-03            NaN            NaN  ...              NaN           NaN
2020-07-02            NaN            NaN  ...              NaN           NaN

[5 rows x 65 columns]
```

2.1.16 Naver Finance Data

Naver Finance provides Korean stock market (KOSPI, KOSDAQ) historical data.

```
In [36]: import pandas_datareader.data as web

In [37]: df = web.DataReader('005930', 'naver', start='2019-09-10', end='2019-10-09')

In [38]: df.head()
Out[38]:
             Open   High    Low  Close    Volume
Date
2019-09-10  47100  47200  46550  47000   9231792
2019-09-11  47300  47400  46800  47150  16141619
2019-09-16  47000  47100  46400  47100  15550926
2019-09-17  47000  47100  46800  46900   7006280
2019-09-18  46900  47700  46800  47700  10413027
```

2.1.17 Yahoo Finance Data

Yahoo Finance provides stock market data

The following endpoints are available:

- `yahoo` - retrieve daily stock prices (high, open, close, volu,e and adjusted close)

- `yahoo-actions` - retrieve historical corporate actions (dividends and stock splits)
- `yahoo-dividends` - retrieve historical dividends

```
In [39]: import pandas_datareader.data as web

In [40]: import pandas as pd

In [41]: import datetime as dt

In [42]: df = web.DataReader('GE', 'yahoo', start='2019-09-10', end='2019-10-09')

In [43]: df.head()
Out[43]:
             High   Low  Open  Close      Volume  Adj Close
Date
2019-09-10  9.27  8.90  8.91   9.14  62617200.0   9.062220
2019-09-11  9.36  9.06  9.15   9.36  57094900.0   9.280347
2019-09-12  9.52  9.22  9.40   9.26  68115100.0   9.181198
2019-09-13  9.45  9.14  9.31   9.34  45589400.0   9.270529
2019-09-16  9.42  9.17  9.30   9.38  45748400.0   9.310231

In [44]: start = dt.datetime(2010, 1, 29)

In [45]: end = dt.datetime.today()

In [46]: actions = web.DataReader('GOOG', 'yahoo-actions', start, end)

In [47]: actions.head()
Out[47]:
            action     value
2015-04-27  SPLIT  0.997262
2014-03-27  SPLIT  0.499500

In [48]: dividends = web.DataReader('IBM', 'yahoo-dividends', start, end)

In [49]: dividends.head()
Out[49]:
              action  value
2021-05-07  DIVIDEND   1.64
2021-02-09  DIVIDEND   1.63
2020-11-09  DIVIDEND   1.63
2020-08-07  DIVIDEND   1.63
2020-05-07  DIVIDEND   1.63
```

2.2 Caching queries

Making the same request repeatedly can use a lot of bandwidth, slow down your code and may result in your IP being banned.

`pandas-datareader` allows you to cache queries using `requests_cache` by passing a `requests_cache.Session` to `DataReader` or `Options` using the `session` parameter.

Below is an example with Yahoo! Finance. The session parameter is implemented for all datareaders.

```
In [1]: import pandas_datareader.data as web

In [2]: from pandas_datareader.yahoo.headers import DEFAULT_HEADERS

In [3]: import datetime

In [4]: import requests_cache

In [5]: expire_after = datetime.timedelta(days=3)

In [6]: session = requests_cache.CachedSession(cache_name='cache', backend='sqlite',␣
↪expire_after=expire_after)

In [7]: session.headers = DEFAULT_HEADERS

In [8]: start = datetime.datetime(2010, 1, 1)

In [9]: end = datetime.datetime(2013, 1, 27)

In [10]: f = web.DataReader("F", 'yahoo', start, end, session=session)

In [11]: f.loc['2010-01-04']
Out[11]:
High         1.028000e+01
Low          1.005000e+01
Open         1.017000e+01
Close        1.028000e+01
Volume       6.085580e+07
Adj Close    6.968545e+00
Name: 2010-01-04 00:00:00, dtype: float64
```

A SQLite file named `cache.sqlite` will be created in the working directory, storing the request until the expiry date.

For additional information on using requests-cache, see the documentation.

2.3 Other Data Sources

Web interfaces are constantly evolving and so there is constant evolution in this space. There are a number of noteworthy Python packages that integrate into the PyData ecosystem that are more narrowly focused than pandas-datareader.

2.3.1 Alpha Vantage

Alpha Vantage provides real time and historical equity data. Users are required to get a free API key before using the API. Documentation is available.

A python package simplifying access is available on github.

2.3.2 Tiingo

Tiingo aims to make high-end financial tools accessible investors. The API is documented. Users are required to get a free API key before using the API.

A python package simplifying access is available on github.

2.3.3 Barchart

Barchart is a data provider covering a wide range of financial data. The free API provides up to two years of historical data.

A python package simplifying access is available on github.

2.3.4 List of Other Sources

Awesome Quant maintains a large list of packages designed to provide access to financial data.

2.4 Data Readers

2.4.1 AlphaVantage

class pandas_datareader.av.forex.**AVForexReader**(*symbols=None*, *retry_count=3*, *pause=0.1*, *session=None*, *api_key=None*)

Returns DataFrame of the Alpha Vantage Foreign Exchange (FX) Exchange Rates data.

New in version 0.7.0.

Parameters

- **symbols** (*string, array-like object (list, tuple, Series)*) – Single currency pair (formatted 'FROM/TO') or list of the same.
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used
- **api_key** (*str, optional*) – Alpha Vantage API key . If not provided the environmental variable ALPHAVANTAGE_API_KEY is read. The API key is *required*.

close()
Close network session

data_key
Key of data returned from Alpha Vantage

default_start_date
Default start date for reader. Defaults to 5 years before current date

function
Alpha Vantage endpoint function

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

class pandas_datareader.av.time_series.**AVTimeSeriesReader**(*symbols=None*, *function='TIME_SERIES_DAILY'*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *chunksize=25*, *api_key=None*)

Returns DataFrame of the Alpha Vantage Stock Time Series endpoints

New in version 0.7.0.

Parameters

- **symbols** (*string*) – Single stock symbol (ticker)
- **start** (*string, int, date, datetime, Timestamp*) – Starting date. Parses many different kind of date representations (e.g., 'JAN-01-2010', '1/1/10', 'Jan, 1, 1980'). Defaults to 20 years before current date.
- **end** (*string, int, date, datetime, Timestamp*) – Ending date
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used
- **api_key** (*str, optional*) – AlphaVantage API key . If not provided the environmental variable ALPHAVANTAGE_API_KEY is read. The API key is *required*.

close()
Close network session

data_key
Key of data returned from Alpha Vantage

default_start_date
Default start date for reader. Defaults to 5 years before current date

function
Alpha Vantage endpoint function

output_size
Used to limit the size of the Alpha Vantage query when possible.

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

class pandas_datareader.av.sector.**AVSectorPerformanceReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *api_key=None*)

Returns DataFrame of the Alpha Vantage Sector Performances SECTOR data.

New in version 0.7.0.

Parameters

- **symbols** (*string, array-like object (list, tuple, Series)*) – Single currency pair (formatted 'FROM/TO') or list of the same.
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used
- **api_key** (*str, optional*) – Alpha Vantage API key . If not provided the environmental variable ALPHAVANTAGE_API_KEY is read. The API key is *required*.

close()
Close network session

data_key
Key of data returned from Alpha Vantage

default_start_date
Default start date for reader. Defaults to 5 years before current date

function
Alpha Vantage endpoint function

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

class pandas_datareader.av.quotes.**AVQuotesReader**(*symbols=None*, *retry_count=3*, *pause=0.1*, *session=None*, *api_key=None*)

Returns DataFrame of Alpha Vantage Realtime Stock quotes for a symbol or list of symbols.

Parameters

- **symbols** (*string, array-like object (list, tuple, Series), or DataFrame*) – Single stock symbol (ticker), array-like object of symbols or DataFrame with index containing stock symbols.
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used

close()
Close network session

data_key
Key of data returned from Alpha Vantage

default_start_date
Default start date for reader. Defaults to 5 years before current date

function
Alpha Vantage endpoint function

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

2.4.2 Federal Reserve Economic Data (FRED)

class pandas_datareader.fred.**FredReader**(*symbols*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *timeout=30*, *session=None*, *freq=None*)

Get data for the given name from the St. Louis FED (FRED).

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

Returns **data** – If multiple names are passed for "series" then the index of the DataFrame is the outer join of the indicies of each series.

Return type DataFrame

url
API URL

2.4.3 Fama-French Data (Ken French's Data Library)

class pandas_datareader.famafrench.**FamaFrenchReader**(*symbols*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *timeout=30*, *session=None*, *freq=None*)

Get data for the given name from the Fama/French data library.

For annual and monthly data, index is a pandas.PeriodIndex, otherwise it's a pandas.DatetimeIndex.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

get_available_datasets()
Get the list of datasets available from the Fama/French data library.

Returns **datasets** – A list of valid inputs for get_data_famafrench

Return type list

params
Parameters to use in API calls

read()
Read data

Returns df – A dictionary of DataFrames. Tables are accessed by integer keys. See df['DESCR'] for a description of the data set.

Return type dict

url
API URL

pandas_datareader.famafrench.**get_available_datasets**(***kwargs*)
Get the list of datasets available from the Fama/French data library.

Parameters **session** (*Session, default None*) – requests.sessions.Session instance to be used

Returns

Return type A list of valid inputs for get_data_famafrench.

2.4.4 Bank of Canada

class pandas_datareader.bankofcanada.**BankOfCanadaReader**(*symbols*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *timeout=30*, *session=None*, *freq=None*)
Get data for the given name from Bank of Canada.

Notes

See Bank of Canada

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

2.4.5 Econdb

class pandas_datareader.econdb.**EcondbReader**(**args*, ***kwargs*)
Get data for the given name from Econdb.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
read one data from specified URL

url
API URL

2.4.6 Enigma

class pandas_datareader.enigma.**EnigmaReader**(*dataset_id=None*, *api_key=None*, *retry_count=5*, *pause=0.75*, *session=None*, *base_url='https://public.enigma.com/api'*)

Collects current snapshot of Enigma data located at the specified data set ID and returns a pandas DataFrame.

Parameters

- **dataset_id** (*str*) – Enigma dataset UUID.
- **api_key** (*str, optional*) – Enigma API key. If not provided, the environmental variable ENIGMA_API_KEY is read.
- **retry_count** (*int, default 5*) – Number of times to retry query request.
- **pause** (*float, default 0.1*) – Time, in seconds, of the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used.
- **base_url** (str, optional (defaults to https://public.enigma.com/api)) – Alternative Enigma endpoint to be used.

Examples

Download current snapshot for the following Florida Inspections Dataset: https://public.enigma.com/datasets/bedaf052-5fcd-4758-8d27-048ce8746c6a

```
>>> import pandas_datareader as pdr
>>> df = pdr.get_data_enigma('bedaf052-5fcd-4758-8d27-048ce8746c6a')
```

In the event that ENIGMA_API_KEY does not exist in your env, the key can be supplied as the second argument or as the keyword argument *api_key*

```
>>> df = EnigmaReader(dataset_id='bedaf052-5fcd-4758-8d27-048ce8746c6a',
...                   api_key='INSERT_API_KEY').read()
```

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

get_current_snapshot_id(*dataset_id*)
Get ID of the most current snapshot of a dataset

get_dataset_metadata(*dataset_id*)
Get the Dataset Model of this EnigmaReader's dataset https://docs.public.enigma.com/resources/dataset/index.html

get_snapshot_export(*snapshot_id*)
Return raw CSV of a dataset

params
Parameters to use in API calls

read()
Read data

url
API URL

2.4.7 Eurostat

class pandas_datareader.eurostat.**EurostatReader**(*symbols*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *timeout=30*, *session=None*, *freq=None*)
Get data for the given name from Eurostat.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

dsd_url
API DSD URL

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

2.4.8 The Investors Exchange (IEX)

class pandas_datareader.iex.daily.**IEXDailyReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *chunksize=25*, *api_key=None*)
Returns DataFrame of historical stock prices from symbols, over date range, start to end. To avoid being penalized by IEX servers, pauses between downloading 'chunks' of symbols can be specified.

Parameters

- **symbols** (*string, array-like object (list, tuple, Series), or DataFrame*) – Single stock symbol (ticker), array-like object of symbols or DataFrame with index containing stock symbols.
- **start** (*string, int, date, datetime, Timestamp*) – Starting date. Parses many different kind of date representations (e.g., 'JAN-01-2010', '1/1/10', 'Jan, 1, 1980'). Defaults to 15 years before current date

- **end** (*string, int, date, datetime, Timestamp*) – Ending date
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **chunksize** (*int, default 25*) – Number of symbols to download consecutively before intiating pause.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used
- **api_key** (*str*) – IEX Cloud Secret Token

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

endpoint
API endpoint

params
Parameters to use in API calls

read()
Read data

url
API URL

class pandas_datareader.iex.market.**MarketReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)

Near real-time traded volume

Notes

Market data is captured by the IEX system between approximately 7:45 a.m. and 5:15 p.m. ET.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

service
Service endpoint

url
API URL

class pandas_datareader.iex.ref.**SymbolsReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)

Symbols available for trading on IEX

Notes

Returns symbols IEX supports for trading. Updated daily as of 7:45 a.m. ET.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

service
Service endpoint

url
API URL

class pandas_datareader.iex.stats.**DailySummaryReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)

Daily statistics from IEX for a day or month

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Unfortunately, IEX's API can only retrieve data one day or one month at a time. Rather than specifying a date range, we will have to run the read function for each date provided.

Returns DataFrame

service
Service endpoint

url
API URL

class pandas_datareader.iex.stats.**MonthlySummaryReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)

Monthly statistics from IEX

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()

Unfortunately, IEX's API can only retrieve data one day or one month at a time. Rather than specifying a date range, we will have to run the read function for each date provided.

Returns DataFrame

service
Service endpoint

url
API URL

class pandas_datareader.iex.stats.**RecordsReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)

Total matched volume information from IEX

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

service
Service endpoint

url
API URL

class pandas_datareader.iex.stats.**RecentReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)

Recent trading volume from IEX

Notes

Returns 6 fields for each day:

- date: refers to the trading day.
- volume: refers to executions received from order routed to away trading centers.
- routedVolume: refers to single counted shares matched from executions on IEX.
- marketShare: refers to IEX's percentage of total US Equity market volume.
- isHalfday: will be true if the trading day is a half day.
- litVolume: refers to the number of lit shares traded on IEX (single-counted).

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

service
Service endpoint

url
API URL

class pandas_datareader.iex.deep.**Deep**(*symbols=None*, *service=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)
Retrieve order book data from IEX

Notes

Real-time depth of book quotations direct from IEX. Returns aggregated size of resting displayed orders at a price and side. Does 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 counted.

Also provides last trade price and size information. Routed executions are not reported.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

service
Service endpoint

url
API URL

class pandas_datareader.iex.tops.**TopsReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)
Near-real time aggregated bid and offer positions

Notes

IEX's aggregated best quoted bid and offer position for all securities on IEX's displayed limit order book.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

service
Service endpoint

url
API URL

class pandas_datareader.iex.tops.**LastReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)
Information of executions on IEX

Notes

Last provides trade data for executions on IEX. Provides last sale price, size and time.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

service
Service endpoint

url
API URL

2.4.9 Moscow Exchange (MOEX)

class pandas_datareader.moex.**MoexReader**(**args*, ***kwargs*)
Returns a DataFrame of historical stock prices from symbols from Moex

Parameters

- **symbols** (*str, an array-like object (list, tuple, Series), or a DataFrame*) – A single stock symbol (secid), an array-like object of symbols or a DataFrame with an index containing stock symbols.
- **start** (*string, int, date, datetime, Timestamp*) – Starting date. Parses many different kind of date representations (e.g., 'JAN-01-2010', '1/1/10', 'Jan, 1, 1980'). Defaults to 20 years before current date.
- **end** (*string, int, date, datetime, Timestamp*) – Ending date
- **retry_count** (*int, default 3*) – The number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **chunksize** (*int, default 25*) – The number of symbols to download consecutively before intiating pause.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used

Notes

To avoid being penalized by Moex servers, pauses more than 0.1s between downloading 'chunks' of symbols can be specified.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data from the primary board for each ticker

read_all_boards()
Read all data from every board for every ticker

url
Return a list of API URLs per symbol

2.4.10 NASDAQ

pandas_datareader.nasdaq_trader.**get_nasdaq_symbols**(*retry_count=3*, *timeout=30*, *pause=None*)

Get the list of all available equity symbols from Nasdaq.

Returns **nasdaq_tickers** – DataFrame with company tickers, names, and other properties.

Return type pandas.DataFrame

2.4.11 Naver Finance

class pandas_datareader.naver.**NaverDailyReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *adjust_price=False*, *ret_index=False*, *chunksize=1*, *interval='d'*, *get_actions=False*, *adjust_dividends=True*)

Fetches daily historical data from Naver Finance.

Parameters

- **symbols** – A single symbol; multiple symbols are not currently supported.
- **adjust_price** – Not implemented
- **interval** – Not implemented
- **adjust_dividends** – Not implemented

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

url
API URL

2.4.12 Organisation for Economic Co-operation and Development (OECD)

class pandas_datareader.oecd.**OECDReader**(*symbols*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *timeout=30*, *session=None*, *freq=None*)

Get data for the given name from OECD.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

2.4.13 Quandl

class pandas_datareader.quandl.**QuandlReader**(*symbols*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *chunksize=25*, *api_key=None*)

Returns DataFrame of historical stock prices from symbol, over date range, start to end.

New in version 0.5.0.

Parameters

- **symbols** (*string*) – Possible formats: 1. DB/SYM: The Quandl 'codes': DB is the database name, SYM is a ticker-symbol-like Quandl abbreviation for a particular security. 2. SYM.CC: SYM is the same symbol and CC is an ISO country code, will try to map to the best single Quandl database for that country. Beware of ambiguous symbols (different securities per country)! Note: Cannot use more than a single string because of the inflexible way the URL is composed of url and _get_params in the superclass
- **start** (*string, int, date, datetime, Timestamp*) – Starting date. Parses many different kind of date representations (e.g., 'JAN-01-2010', '1/1/10', 'Jan, 1, 1980'). Defaults to 20 years before current date.
- **end** (*string, int, date, datetime, Timestamp*) – Ending date
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **chunksize** (*int, default 25*) – Number of symbols to download consecutively before intiating pause.

- **session** (*Session, default None*) – requests.sessions.Session instance to be used
- **api_key** (*str, optional*) – Quandl API key . If not provided the environmental variable QUANDL_API_KEY is read. The API key is *required*.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

url
API URL

2.4.14 Stooq.com

class pandas_datareader.stooq.**StooqDailyReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *chunksize=25*)
Returns DataFrame/dict of Dataframes of historical stock prices from symbols, over date range, start to end.

Parameters

- **symbols** (*string, array-like object (list, tuple, Series), or DataFrame*) – Single stock symbol (ticker), array-like object of symbols or DataFrame with index containing stock symbols.
- **start** (*string, int, date, datetime, Timestamp*) – Starting date. Parses many different kind of date representations (e.g., 'JAN-01-2010', '1/1/10', 'Jan, 1, 1980'). Defaults to 20 years before current date.
- **end** (*string, int, date, datetime, Timestamp*) – Ending date
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **chunksize** (*int, default 25*) – Number of symbols to download consecutively before initiating pause.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used
- **freq** (*string, d, w, m ,q, y for daily, weekly, monthly, quarterly, yearly*) –

Notes

See Stooq

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

url
API URL

2.4.15 Tiingo

class pandas_datareader.tiingo.**TiingoDailyReader**(*symbols*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *timeout=30*, *session=None*, *freq=None*, *api_key=None*)

Historical daily data from Tiingo on equities, ETFs and mutual funds

Parameters

- **symbols** (*{str, List[str]}*) – String symbol or list of symbols
- **start** (*string, int, date, datetime, Timestamp*) – Starting date, timestamp. Parses many different kind of date representations (e.g., 'JAN-01-2010', '1/1/10', 'Jan, 1, 1980'). Default starting date is 5 years before current date.
- **end** (*string, int, date, datetime, Timestamp*) – Ending date, timestamp. Same format as starting date.
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*float, default 0.1*) – Time, in seconds, of the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used
- **freq** (*{str, None}*) – Not used.
- **api_key** (*str, optional*) – Tiingo API key . If not provided the environmental variable TIINGO_API_KEY is read. The API key is *required*.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

class pandas_datareader.tiingo.**TiingoQuoteReader**(*symbols*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *timeout=30*, *session=None*, *freq=None*, *api_key=None*)

Read quotes (latest prices) from Tiingo

Parameters

- **symbols** (*{str, List[str]}*) – String symbol or list of symbols

- **start** (*string, int, date, datetime, Timestamp*) – Not used.
- **end** (*string, int, date, datetime, Timestamp*) – Not used.
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*float, default 0.1*) – Time, in seconds, of the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used
- **freq** (*{str, None}*) – Not used.
- **api_key** (*str, optional*) – Tiingo API key . If not provided the environmental variable TIINGO_API_KEY is read. The API key is *required*.

Notes

This is a special case of the daily reader which automatically selected the latest data available for each symbol.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

class pandas_datareader.tiingo.**TiingoMetaDataReader**(*symbols*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *timeout=30*, *session=None*, *freq=None*, *api_key=None*)

Read metadata about symbols from Tiingo

Parameters

- **symbols** (*{str, List[str]}*) – String symbol or list of symbols
- **start** (*string, int, date, datetime, Timestamp*) – Not used.
- **end** (*string, int, date, datetime, Timestamp*) – Not used.
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*float, default 0.1*) – Time, in seconds, of the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used
- **freq** (*{str, None}*) – Not used.
- **api_key** (*str, optional*) – Tiingo API key . If not provided the environmental variable TIINGO_API_KEY is read. The API key is *required*.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

pandas_datareader.tiingo.**get_tiingo_symbols**()
Get the set of stock symbols supported by Tiingo

Returns **symbols** – DataFrame with symbols (ticker), exchange, asset type, currency and start and end dates

Return type DataFrame

Notes

Reads https://apimedia.tiingo.com/docs/tiingo/daily/supported_tickers.zip

2.4.16 Thrift Savings Plan (TSP)

class pandas_datareader.tsp.**TSPReader**(*symbols=frozenset({'L 2040', 'C Fund', 'L 2030', 'L Income', 'S Fund', 'L 2025', 'I Fund', 'L 2045', 'L 2035', 'L 2060', 'L 2065', 'L 2055', 'F Fund', 'L 2050', 'G Fund'})*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)
Returns DataFrame of historical TSP fund prices from symbols, over date range, start to end.

Parameters

- **symbols** (*str, array-like object (list, tuple, Series), or DataFrame*) – Single stock symbol (ticker), array-like object of symbols or DataFrame with index containing stock symbols.
- **start** (*string, int, date, datetime, Timestamp*) – Starting date. Parses many different kind of date representations (e.g., 'JAN-01-2010', '1/1/10', 'Jan, 1, 1980'). Defaults to 20 years before current date.
- **end** (*string, int, date, datetime, Timestamp*) – Ending date
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
read one data from specified URL

url
API URL

2.4.17 World Bank

class pandas_datareader.wb.**WorldBankReader**(*symbols=None*, *countries=None*, *start=None*, *end=None*, *freq=None*, *retry_count=3*, *pause=0.1*, *session=None*, *errors='warn'*)

Download data series from the World Bank's World Development Indicators

Parameters

- **symbols** (*WorldBank indicator string or list of strings*) – taken from the `id` field in `WDIsearch()`
- **countries** (*string or list of strings.*) – `all` downloads data for all countries 2 or 3 character ISO country codes select individual countries (e.g.''US'',''CA'') or (e.g.''USA'',''CAN''). The codes can be mixed. The two ISO lists of countries, provided by wikipedia, are hardcoded into pandas as of 11/10/2014.
- **start** (*string, int, date, datetime, Timestamp*) – First year of the data series. Month and day are ignored.
- **end** (*string, int, date, datetime, Timestamp*) – Last year of the data series (inclusive). Month and day are ignored.
- **errors** (*str {'ignore', 'warn', 'raise'}, default 'warn'*) – Country codes are validated against a hardcoded list. This controls the outcome of that validation, and attempts to also apply to the results from world bank. errors='raise', will raise a ValueError on a bad country code.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

get_countries()
Query information about countries

Notes

Provides information such as:

- country code
- region
- income level
- capital city
- latitude
- and longitude

get_indicators()
Download information about all World Bank data series

params
Parameters to use in API calls

read()
Read data

search(*string='gdp.*capi'*, *field='name'*, *case=False*)
Search available data series from the world bank

Parameters

- **string** (*string*) – regular expression
- **field** (*string*) – id, name, source, sourceNote, sourceOrganization, topics See notes below
- **case** (*bool*) – case sensitive search?

Notes

The first time this function is run it will download and cache the full list of available series. Depending on the speed of your network connection, this can take time. Subsequent searches will use the cached copy, so they should be much faster.

id : Data series indicator (for use with the `indicator` argument of `WDI()`) e.g. NY.GNS.ICTR.GN.ZS" name: Short description of the data series source: Data collection project sourceOrganization: Data collection organization note: sourceNote: topics:

url
API URL

pandas_datareader.wb.**download**(*country=None*, *indicator=None*, *start=2003*, *end=2005*, *freq=None*, *errors='warn'*, ***kwargs*)
Download data series from the World Bank's World Development Indicators

Parameters

- **indicator** (*string or list of strings*) – taken from the `id` field in `WDIsearch()`
- **country** (*string or list of strings.*) – `all` downloads data for all countries 2 or 3 character ISO country codes select individual countries (e.g.''US'',''CA'') or (e.g.''USA'',''CAN''). The codes can be mixed.

  The two ISO lists of countries, provided by wikipedia, are hardcoded into pandas as of 11/10/2014.
- **start** (*int*) – First year of the data series
- **end** (*int*) – Last year of the data series (inclusive)
- **freq** (*str*) – frequency or periodicity of the data to be retrieved (e.g. 'M' for monthly, 'Q' for quarterly, and 'A' for annual). None defaults to annual.
- **errors** (*str {'ignore', 'warn', 'raise'}, default 'warn'*) – Country codes are validated against a hardcoded list. This controls the outcome of that validation, and attempts to also apply to the results from world bank. errors='raise', will raise a ValueError on a bad country code.
- **kwargs** – keywords passed to WorldBankReader

Returns **data** – DataFrame with columns country, year, indicator value

Return type DataFrame

pandas_datareader.wb.**get_countries**(***kwargs*)
Query information about countries

Provides information such as: country code, region, income level, capital city, latitude, and longitude

Parameters **kwargs** – keywords passed to WorldBankReader

pandas_datareader.wb.**get_indicators**(***kwargs*)
Download information about all World Bank data series

Parameters **kwargs** – keywords passed to WorldBankReader

pandas_datareader.wb.**search**(*string='gdp.*capi'*, *field='name'*, *case=False*, ***kwargs*)
Search available data series from the world bank

Parameters

- **string** (*string*) – regular expression
- **field** (*string*) – id, name, source, sourceNote, sourceOrganization, topics. See notes
- **case** (*bool*) – case sensitive search?
- **kwargs** – keywords passed to WorldBankReader

Notes

The first time this function is run it will download and cache the full list of available series. Depending on the speed of your network connection, this can take time. Subsequent searches will use the cached copy, so they should be much faster.

id : Data series indicator (for use with the `indicator` argument of `WDI()`) e.g. NY.GNS.ICTR.GN.ZS"

- name: Short description of the data series
- source: Data collection project
- sourceOrganization: Data collection organization
- note:
- sourceNote:
- topics:

2.4.18 Yahoo Finance

class pandas_datareader.yahoo.daily.**YahooDailyReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *adjust_price=False*, *ret_index=False*, *chunksize=1*, *interval='d'*, *get_actions=False*, *adjust_dividends=True*)
Returns DataFrame of with historical over date range, start to end. To avoid being penalized by Yahoo! Finance servers, pauses between downloading 'chunks' of symbols can be specified.

Parameters

- **symbols** (*string, array-like object (list, tuple, Series), or DataFrame*) – Single stock symbol (ticker), array-like object of symbols or DataFrame with index containing stock symbols.
- **start** (*string, int, date, datetime, Timestamp*) – Starting date. Parses many different kind of date representations (e.g., 'JAN-01-2010', '1/1/10', 'Jan, 1, 1980'). Defaults to 5 years before current date.
- **end** (*string, int, date, datetime, Timestamp*) – Ending date
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used. Passing a session is an advanced usage and you must set any required headers in the session directly.
- **adjust_price** (*bool, default False*) – If True, adjusts all prices in hist_data ('Open', 'High', 'Low', 'Close') based on 'Adj Close' price. Adds 'Adj_Ratio' column and drops 'Adj Close'.
- **ret_index** (*bool, default False*) – If True, includes a simple return index 'Ret_Index' in hist_data.
- **chunksize** (*int, default 25*) – Number of symbols to download consecutively before intiating pause.
- **interval** (*string, default 'd'*) – Time interval code, valid values are 'd' for daily, 'w' for weekly, 'm' for monthly.
- **get_actions** (*bool, default False*) – If True, adds Dividend and Split columns to dataframe.
- **adjust_dividends** (*bool, default true*) – If True, adjusts dividends for splits.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

url
API URL

class pandas_datareader.yahoo.fx.**YahooFXReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *adjust_price=False*, *ret_index=False*, *chunksize=1*, *interval='d'*, *get_actions=False*, *adjust_dividends=True*)

Returns DataFrame of historical prices for currencies

Parameters

- **symbols** (*string, array-like object (list, tuple, Series), or DataFrame*) – Single stock symbol (ticker), array-like object of symbols or DataFrame with index containing stock symbols.
- **start** (*string, int, date, datetime, Timestamp*) – Starting date, timestamp. Parses many different kind of date representations (e.g., 'JAN-01-2010', '1/1/10', 'Jan, 1, 1980'). Defaults to '1/1/2010'.
- **end** (*string, int, date, datetime, Timestamp*) – Ending date, timestamp. Same format as starting date. Defaults to today.
- **retry_count** (*int, default 3*) – Number of times to retry query request.
- **pause** (*int, default 0.1*) – Time, in seconds, to pause between consecutive queries of chunks. If single value given for symbol, represents the pause between retries.
- **session** (*Session, default None*) – requests.sessions.Session instance to be used
- **chunksize** (*int, default 25 (NOT IMPLEMENTED)*) – Number of symbols to download consecutively before intiating pause.
- **interval** (*string, default '1d'*) –

  Valid values are '1d', '5d', '1mo', '3mo', '6mo', '1y', '2y', '5y', '10y', 'ytd', 'max'

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

url
API URL

class pandas_datareader.yahoo.options.**Options**(*symbol*, *session=None*)
Experimental This class fetches call/put data for a given stock/expiry month.

It is instantiated with a string representing the ticker symbol.

The class has the following methods: get_options_data(month, year, expiry) get_call_data(month, year, expiry) get_put_data(month, year, expiry) get_near_stock_price(opt_frame, above_below) get_all_data(call, put) get_forward_data(months, call, put) (deprecated)

Examples

Instantiate object with ticker >>> aapl = Options('aapl')

Fetch next expiry call data >>> calls = aapl.get_call_data()

Can now access aapl.calls instance variable >>> aapl.calls

Fetch next expiry put data >>> puts = aapl.get_put_data()

Can now access aapl.puts instance variable >>> aapl.puts

cut down the call data to be 3 below and 3 above the stock price. >>> cut_calls = aapl.get_near_stock_price(call=True, above_below=3)

Fetch call and put data with expiry from now to 8 months out >>> forward_data = aapl.get_forward_data(8, call=True, put=True)

Fetch all call and put data >>> all_data = aapl.get_all_data()

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

expiry_dates
Returns a list of available expiry dates

get_all_data(*call=True*, *put=True*)
Experimental Gets either call, put, or both data for all available months starting in the current month.

Parameters

- **call** (*bool, optional (default=True)*) – Whether or not to collect data for call options
- **put** (*bool, optional (default=True)*) – Whether or not to collect data for put options.

Returns

- *pandas.DataFrame* – A DataFrame with requested options data.

  Index: Strike: Option strike, int Expiry: Option expiry, Timestamp Type: Call or Put, string Symbol: Option symbol as reported on Yahoo, string

  Columns: Last: Last option price, float Chg: Change from prior day, float PctChg: Change from prior day in percent, float Bid: Bid price, float Ask: Ask price, float Vol: Volume traded, int64 Open_Int: Open interest, int64 IsNonstandard: True if the the deliverable is not 100 shares,

  otherwise false

  Underlying: Ticker of the underlying security, string Underlying_Price: Price of the underlying security, float64 Quote_Time: Time of the quote, Timestamp Last_Trade_Date: Time of the last trade for this expiry

  and strike, Timestamp

  IV: Implied volatility, float JSON: Parsed json object, json

  Useful to extract other returned key/value pairs as needed

- **Note** (*Format of returned DataFrame is dependent*) – on Yahoo and may change.

get_call_data(*month=None*, *year=None*, *expiry=None*)
Experimental Gets call/put data for the stock with the expiration data in the given month and year

Parameters

- **month** (*number, int, optional(default=None)*) – The month the options expire. This should be either 1 or 2 digits.
- **year** (*number, int, optional(default=None)*) – The year the options expire. This should be a 4 digit int.
- **expiry** (*date-like or convertible or*) – list-like object, optional (default=None) The date (or dates) when options expire (defaults to current month)

Returns

call_data – A DataFrame with requested options data.

Index: Strike: Option strike, int Expiry: Option expiry, Timestamp Type: Call or Put, string Symbol: Option symbol as reported on Yahoo, string

Columns: Last: Last option price, float Chg: Change from prior day, float PctChg: Change from prior day in percent, float Bid: Bid price, float Ask: Ask price, float Vol: Volume traded, int64 Open_Int: Open interest, int64 IsNonstandard: True if the the deliverable is not 100 shares,

otherwise false

Underlying: Ticker of the underlying security, string Underlying_Price: Price of the underlying security, float64 Quote_Time: Time of the quote, Timestamp Last_Trade_Date: Time of the last trade for this expiry

and strike, Timestamp

IV: Implied volatility, float JSON: Parsed json object, json

Useful to extract other returned key/value pairs as needed

Return type pandas.DataFrame

Notes

Note: Format of returned DataFrame is dependent on Yahoo and may change.

When called, this function will add instance variables named calls and puts. See the following example:

```
>>> aapl = Options('aapl', 'yahoo')  # Create object
>>> aapl.calls  # will give an AttributeError
>>> aapl.get_call_data()  # Get data and set ivars
>>> aapl.calls  # Doesn't throw AttributeError
```

Also note that aapl.calls will always be the calls for the next expiry. If the user calls this method with a different month or year, the ivar will be named callsYYMMDD where YY, MM and DD are, respectively, two digit representations of the year, month and day for the expiry of the options.

get_forward_data(*months*, *call=True*, *put=False*, *near=False*, *above_below=2*)
Experimental Gets either call, put, or both data for months starting in the current month and going out in the future a specified amount of time.

Parameters

- **months** (*number, int*) – How many months to go out in the collection of the data. This is inclusive.
- **call** (*bool, optional (default=True)*) – Whether or not to collect data for call options
- **put** (*bool, optional (default=False)*) – Whether or not to collect data for put options.
- **near** (*bool, optional (default=False)*) – Whether this function should get only the data near the current stock price. Uses Options.get_near_stock_price
- **above_below** (*number, int, optional (default=2)*) – The number of strike prices above and below the stock price that should be taken if the near option is set to True

Returns

A DataFrame with requested options data.

Index: Strike: Option strike, int Expiry: Option expiry, Timestamp Type: Call or Put, string Symbol: Option symbol as reported on Yahoo, string

Columns: Last: Last option price, float Chg: Change from prior day, float PctChg: Change from prior day in percent, float Bid: Bid price, float Ask: Ask price, float Vol: Volume traded, int64 Open_Int: Open interest, int64 IsNonstandard: True if the the deliverable is not 100 shares,

otherwise false

Underlying: Ticker of the underlying security, string Underlying_Price: Price of the underlying security, float64 Quote_Time: Time of the quote, Timestamp Last_Trade_Date: Time of the last trade for this expiry

and strike, Timestamp

IV: Implied volatility, float JSON: Parsed json object, json

Useful to extract other returned key/value pairs as needed

Note: Format of returned DataFrame is dependent on Yahoo and may change.

Return type pandas.DataFrame

get_near_stock_price(*above_below=2*, *call=True*, *put=False*, *month=None*, *year=None*, *expiry=None*)

Experimental Returns a DataFrame of options that are near the current stock price.

Parameters

- **above_below** (*number, int, optional (default=2)*) – The number of strike prices above and below the stock price that should be taken
- **call** (*bool*) – Tells the function whether or not it should be using calls
- **put** (*bool*) – Tells the function weather or not it should be using puts
- **month** (*number, int, optional(default=None)*) – The month the options expire. This should be either 1 or 2 digits.
- **year** (*number, int, optional(default=None)*) – The year the options expire. This should be a 4 digit int.
- **expiry** (*date-like or convertible or list-like object,*) – optional (default=None) The date (or dates) when options expire (defaults to current month)

Returns

chopped – The resultant DataFrame chopped down to be 2 * above_below + 1 rows desired. If there isn't data as far out as the user has asked for then

Index: Strike: Option strike, int Expiry: Option expiry, Timestamp Type: Call or Put, string Symbol: Option symbol as reported on Yahoo, string

Columns: Last: Last option price, float Chg: Change from prior day, float PctChg: Change from prior day in percent, float Bid: Bid price, float Ask: Ask price, float Vol: Volume traded, int64 Open_Int: Open interest, int64 IsNonstandard: True if the the deliverable is not 100 shares,

otherwise false

Underlying: Ticker of the underlying security, string Underlying_Price: Price of the underlying security, float64 Quote_Time: Time of the quote, Timestamp Last_Trade_Date: Time of the last trade for this expiry

and strike, Timestamp

IV: Implied volatility, float JSON: Parsed json object, json

Useful to extract other returned key/value pairs as needed

Note: Format of returned DataFrame is dependent on Yahoo and may change.

Return type DataFrame

get_options_data(*month=None*, *year=None*, *expiry=None*)

Experimental Gets call/put data for the stock with the expiration data in the given month and year

Parameters

- **month** (*number, int, optional(default=None)*) – The month the options expire. This should be either 1 or 2 digits.
- **year** (*number, int, optional(default=None)*) – The year the options expire. This should be a 4 digit int.
- **expiry** (*date-like or convertible or*) – list-like object, optional (default=None) The date (or dates) when options expire (defaults to current month)

Returns

A DataFrame with requested options data.

Index: Strike: Option strike, int Expiry: Option expiry, Timestamp Type: Call or Put, string Symbol: Option symbol as reported on Yahoo, string

Columns: Last: Last option price, float Chg: Change from prior day, float PctChg: Change from prior day in percent, float Bid: Bid price, float Ask: Ask price, float Vol: Volume traded, int64 Open_Int: Open interest, int64 IsNonstandard: True if the the deliverable is not 100 shares,

otherwise False

Underlying: Ticker of the underlying security, string Underlying_Price: Price of the underlying security, float64 Quote_Time: Time of the quote, Timestamp Last_Trade_Date: Time of the last trade for this expiry

and strike, Timestamp

IV: Implied volatility, float JSON: Parsed json object, json

Useful to extract other returned key/value pairs as needed

Return type pandas.DataFrame

Notes

Note: Format of returned DataFrame is dependent on Yahoo and may change.

When called, this function will add instance variables named calls and puts. See the following example:

```
>>> aapl = Options('aapl', 'yahoo')  # Create object
>>> aapl.calls  # will give an AttributeError
>>> aapl.get_options()  # Get data and set ivars
>>> aapl.calls  # Doesn't throw AttributeError
```

Also note that aapl.calls and appl.puts will always be the calls and puts for the next expiry. If the user calls this method with a different expiry, the ivar will be named callsYYMMDD or putsYYMMDD, where YY, MM and DD are, respectively, two digit representations of the year, month and day for the expiry of the options.

get_put_data(*month=None*, *year=None*, *expiry=None*)

Experimental Gets put data for the stock with the expiration data in the given month and year

Parameters

- **month** (*number, int, optional(default=None)*) – The month the options expire. This should be either 1 or 2 digits.
- **year** (*number, int, optional(default=None)*) – The year the options expire. This should be a 4 digit int.
- **expiry** (*date-like or convertible or*) – list-like object, optional (default=None) The date (or dates) when options expire (defaults to current month)

Returns

put_data – A DataFrame with requested options data.

Index: Strike: Option strike, int Expiry: Option expiry, Timestamp Type: Call or Put, string Symbol: Option symbol as reported on Yahoo, string

Columns: Last: Last option price, float Chg: Change from prior day, float PctChg: Change from prior day in percent, float Bid: Bid price, float Ask: Ask price, float Vol: Volume traded, int64 Open_Int: Open interest, int64 IsNonstandard: True if the the deliverable is not 100 shares,

otherwise false

Underlying: Ticker of the underlying security, string Underlying_Price: Price of the underlying security, float64 Quote_Time: Time of the quote, Timestamp Last_Trade_Date: Time of the last trade for this expiry

and strike, Timestamp

IV: Implied volatility, float JSON: Parsed json object, json

Useful to extract other returned key/value pairs as needed

Return type pandas.DataFrame

Notes

Note: Format of returned DataFrame is dependent on Yahoo and may change.

When called, this function will add instance variables named puts. See the following example:

```
>>> aapl = Options('aapl')  # Create object
>>> aapl.puts  # will give an AttributeError
>>> aapl.get_put_data()  # Get data and set ivars
>>> aapl.puts  # Doesn't throw AttributeError
```

```
return self.__setattr__(self, str(str(x) + str(y)))
```

Also note that aapl.puts will always be the puts for the next expiry. If the user calls this method with a different month or year, the ivar will be named putsYYMMDD where YY, MM and DD are, respectively, two digit representations of the year, month and day for the expiry of the options.

params
Parameters to use in API calls

quote_time
Returns the quote time.

read()
Read data from connector

underlying_price
Returns the underlying price.

url
API URL

class pandas_datareader.yahoo.quotes.**YahooQuotesReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*)

Get current yahoo quote

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params(*symbol*)
Parameters to use in API calls

read()
Read data from connector

url
API URL

class pandas_datareader.yahoo.components.**_get_data**
Returns DataFrame containing list of component information for index represented in idx_sym from yahoo. Includes component symbol (ticker), exchange, and name.

Parameters **idx_sym** (*str*) – Stock index symbol Examples: '^DJI' (Dow Jones Industrial Average) '^NYA' (NYSE Composite) '^IXIC' (NASDAQ Composite)

See: http://finance.yahoo.com/indices for other index symbols

Returns **idx_df**

Return type DataFrame

class pandas_datareader.yahoo.actions.**YahooActionReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *adjust_price=False*, *ret_index=False*, *chunksize=1*, *interval='d'*, *get_actions=False*, *adjust_dividends=True*)

Returns DataFrame of historical corporate actions (dividends and stock splits) from symbols, over date range, start to end. All dates in the resulting DataFrame correspond with dividend and stock split ex-dates.

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

url
API URL

class pandas_datareader.yahoo.actions.**YahooDivReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *adjust_price=False*, *ret_index=False*, *chunksize=1*, *interval='d'*, *get_actions=False*, *adjust_dividends=True*)

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

url
API URL

class pandas_datareader.yahoo.actions.**YahooSplitReader**(*symbols=None*, *start=None*, *end=None*, *retry_count=3*, *pause=0.1*, *session=None*, *adjust_price=False*, *ret_index=False*, *chunksize=1*, *interval='d'*, *get_actions=False*, *adjust_dividends=True*)

close()
Close network session

default_start_date
Default start date for reader. Defaults to 5 years before current date

params
Parameters to use in API calls

read()
Read data

url
API URL

2.5 What's New

These are new features and improvements of note in each release.

2.5.1 v0.10.0 (July 11, 2021)

Highlights include:

Bug Fixes

- Fixed Yahoo readers which now require headers
- Fixed other reader
- Improved compatibility with pandas

Contributors

Thanks to all of the contributors for the 0.10.0 release (based on git log):

- Igor Molnar
- Kevin Sheppard
- galashour
- David Kim
- Kim Philipp Jablonski
- Tim Gates
- Jeff Hale
- Lukas Halim
- Simon Garisch
- Dmitry Alekseev

These lists of names are automatically generated based on git log, and may not be complete.

2.5.2 v0.9.1 (July 26, 2020)

Highlights include:

What's new in v0.9.1

- *Enhancements*

Enhancements

- MOEX now returns data only from primary trading boards for each ticker (GH811, GH748)
- Added read_all_boards() method for MOEX that returns data from every trading board (ver. 0.9.0 behaviour)
- Docs for MOEX reedited

Bug Fixes

- Fixed broken links on the github ussies on "What's New" docs pages

Contributors

- Dmitry Alekseev

2.5.3 v0.9.0 (July 10, 2020)

Highlights include:

What's new in v0.9.0

Enhancements

- Added a timeout parameter to prevent infinite hangs (GH790)
- Added AlphaVantage endpoint to get historical currency exchange rates (GH764)
- Improved logging when rate limited (GH745)
- Added daily historical data from Naver Finance (GH722)
- Added the method `test` to the package to simplify running tests from an installation using

```
python -c "import pandas_datareader; pandas_datareader.test()"
```

Backwards incompatible API changes

- Dropped support for Python 2.7. The minimum python version is now Python 3.6.
- Removed Robinhood which no longer exists
- Immediately deprecated AlphaVantage quote reader which used an end point that has been retired
- Immediately deprecated Enigma which has substantially changed their business model and API
- Immediately deprecated old symbol names for TSP to reflect the new website API

Bug Fixes

- Fix Yahoo Splits for change in format (GH756)
- Fixed reading bond yields from Stooq (GH752)
- Update TSP web scraper to new site (GH740)
- Fixed EconDB reader to use session (GH737)
- Fix reading futures data from Stooq (GH718)
- Correct NASDAQ symbols fields link (GH715)
- Fix Yahoo! actions bug due to change in split format (GH755)
- Fix FutureWarning from pandas import (GH762)

Contributors

- Spencer Adams
- Jesse Aldridge
- Sumin Byeon
- Patrick Collins
- David Cottrell
- Greg
- Matthew Hall
- Kirill Ikonnikov
- Gábor Lipták
- Kevin Sheppard
- David Stephens
- Eldon Allred

2.5.4 v0.8.0 (September 22, 2019)

Highlights include:

- A new connector for Econdb was introduced. Econdb provides aggregated economic data from 90+ official statistical agencies (GH615)
- Migrated IEX readers to IEX Cloud. All readers now require an API token (`IEX_API_KEY`) (GH638)

- Removal of Google Finance and Morningstar, which were deprecated in 0.7.0
- Immediate deprecation of Robinhood for quotes and historical data. Robinhood ended support for these endpoints in 1/2019

What's new in v0.8.0

Enhancements

- Added Tiingo IEX Historical reader. (GH619)
- Added support for Alpha Vantage intraday time series prices (GH631)
- Up to 15 years of historical prices from IEX with new platform, IEX Cloud
- Added testing on Python 3.7 (GH667)
- Allow IEX to read less than 1 year of data (GH649)
- Allow data download from Poland using stooq (GH597)
- All time series readers now use a rolling default starting date (most are 5 years before the current date. Intraday readers are 3-5 days from the current date)

Backwards incompatible API changes

- Immediate deprecation of Robinhood for quotes and historical data. Robinhood ended support for these endpoints in 1/2019. The Robinhood quotes and daily readers will raise an `ImmediateDeprecationError` when called.
- Usage of all IEX readers requires an IEX Cloud API token, which can be passed as a parameter or stored in the environment variable `IEX_API_TOKEN`
- Deprecated `access_key` in favor of `api_key` in `DataReader`. (GH693)

Bug Fixes

- Fix Yahoo! actions issue where dividends are adjusted twice as a result of a change to the Yahoo! API. (GH583)
- Fix AlphaVantage time series data ordering after provider switch to descending order (maintains ascending order for consistency). (GH662)
- Refactored compatibility library to be independent of pandas version.
- Fixed quarter value handling in JSDMX and OECD. (GH685)
- Fixed a bug in `base` so that the reader does not error when response.encoding is `None`. (GH674)
- Correct EcondbReader's API URL format. (GH670)
- Fix eurostat URL. (GH669)

- Adjust Alphavantage time series reader to account for descending ordering. (GH666)
- Fix bug in downloading index historical constituents. (GH591)
- Fix a bug that occurs when an endpoint returns has no data for a date range. (GH640)

Contributors

- Peiji Chen
- EconDB
- Roger Erens
- Nikhilesh Koshti
- Gábor Lipták
- Addison Lynch
- Rahim Nathwani
- Chuk Orakwue
- Raffaele Sandrini
- Felipe S. S. Schneider
- Kevin Sheppard
- Tony Shouse
- David Stephens

2.5.5 v0.7.0 (September 11, 2018)

Warning: Google finance and Morningstar for historical price data have been immediately deprecated.

Highlights include:

- Immediate deprecation of Google finance and Morningstar for historical price data, as these API endpoints are no longer supported by their respective providers. Alternate methods are welcome via pull requests, as PDR would like to restore these features.
- Removal of EDGAR, which was deprecated in v0.6.0.

What's new in v0.7.0

- *Enhancements*
- *Backwards incompatible API changes*
- *Bug Fixes*

Enhancements

- A new data connector for data provided by Alpha Vantage was introduced to obtain Foreign Exchange (FX) data. (GH389)
- A new data connector for data provided by Alpha Vantage was introduced to obtain historical time series data. (GH389)
- **A new data connector for data provided by** Alpha Vantage was introduced to obtain sector performance data, accessed through the top-level function `get_sector_performance_av`. (GH389)
- A new data connector for data provided by Alpha Vantage was introduced to obtain real-time Batch Stock Quotes through the top-level function `get_quote_av`. (GH389)
- MOEX data connector now supports multiple symbols in constructor. (GH562)

Backwards incompatible API changes

- Deprecation of Google finance daily reader. Google retired the remaining financial data end point in June 2018. It is not possible to reliably retrieve historical price data without this endpoint. The Google daily reader will raise an *ImmediateDeprecationError* when called.
- Deprecation of Morningstar daily reader. Morningstar ended support for the historical price data endpoint in July 2018. It is not possible to retrieve historical price data without this endpoint. The Morningstar daily reader will raise an *ImmediateDeprecationError* when called.
- When requesting multiple symbols from a DailyReader (ex: google, yahoo, IEX) a MultiIndex DataFrame is now returned. Previously Panel or dict of DataFrames were returned. (GH297).

Bug Fixes

- Fixed import of pandas.compat (GH657)
- Added support for passing the API KEY to QuandlReader either directly or by setting the environmental variable QUANDL_API_KEY (GH485).
- Added support for optionally passing a custom base_url to the EnigmaReader (GH499).
- Fix Yahoo! price data (GH498).
- Added back support for Yahoo! price, dividends, and splits data for stocks and currency pairs (GH487).
- Add *is_list_like* to compatibility layer to avoid failure on pandas >= 0.23 (GH520).
- Fixed Yahoo! time offset (GH487).
- Fix Yahoo! quote reader (GH540).
- Remove import of deprecated *tm.get_data_path* (GH566)
- Allow full usage of stooq url parameters.
- Removed unused requests-file and requests-ftp dependencies.
- Fix Yahoo! actions issue where the default reporting adjusts dividends. The unadjusted dividends may lack precision due to accumulated numerical error when converting adjusted to the original dividend amount. (GH495)

2.5.6 v0.6.0 (January 24, 2018)

This is a major release from 0.5.0. We recommend that all users upgrade.

Warning: Yahoo!, Google Options, Google Quotes and EDGAR have been immediately deprecated.

Note: Google finance is still functioning for historical price data, although there are frequent reports of failures. Failure is frequently encountered when bulk downloading historical price data.

Highlights include:

- Immediate deprecation of Yahoo!, Google Options and Quotes and EDGAR. The end points behind these APIs have radically changed and the existing readers require complete rewrites. In the case of most Yahoo! data the endpoints have been removed. PDR would like to restore these features, and pull requests are welcome.
- A new connector for Tiingo was introduced. Tiingo provides historical end-of-day data for a large set of equities, ETFs and mutual funds. Free registration is required to get an API key (GH478).
- A new connector for Robinhood was introduced. This provides up to 1 year of historical end-of-day data. It also provides near real-time quotes. (GH477).
- A new connector for Morningstar Open, High, Low, Close and Volume was introduced (GH467)
- A new connector for IEX daily price data was introduced (GH465).
- A new connector for IEX the majority of the IEX API was introduced (GH446).
- A new data connector for stock index data provided by Stooq was introduced (GH447).
- A new data connector for data provided by the Bank of Canada was introduced (GH440).
- A new data connector for data provided by Moscow Exchange (MOEX) introduced (GH381).

What's new in v0.6.0

Enhancements

- A new data connector for data provided by the Bank of Canada was introduced. (GH440)
- A new data connector for stock index data provided by Stooq was introduced. (GH447)
- A new connector for IEX the majority of the IEX API was introduced (GH446).
- A new connector for IEX daily price data was introduced (GH465).
- A new data connector for stock pricing data provided by Morningstar was introduced. (GH467)
- A new data connector for stock pricing data provided by Robinhood was introduced. (GH477)
- A new data connector for stock pricing data provided by Tiingo was introduced. (GH478)

- A new data connector for data provided by Moscow Exchange was introduced. (GH381).

Backwards incompatible API changes

- Deprecation of Yahoo readers. Yahoo! retired the financial data end points in late 2017. It is not possible to reliably retrieve data from Yahoo! without these endpoints. The Yahoo! readers have been immediately deprecated and will raise an *ImmediateDeprecationError* when called.
- Deprecation of EDGAR readers. EDGAR substantially altered their API. The EDGAR readers have been immediately deprecated and will raise an *ImmediateDeprecationError* when called.
- Google finance data will raise an *UnstableAPIWarning* when first called. Google has also altered their API in a way that makes reading data unreliable. It many call it works. However it also regularly fails, especially when used for bulk downloading. Google may be removed in the future.

Bug Fixes

- *freq* parameter was added to the WorldBank connector to address a limitation (GH198, GH449).
- The Enigma data connector was updated to the latest API (GH380).
- The Google finance endpoint was updated to the latest value (GH404).
- Tne end point for FRED was updated to the latest values (GH436).
- Tne end point for WorldBank was updated to the latest values (GH456).

Other Changes

- The minimum tested pandas version was increased to 0.19.2 (GH441).
- Added versioneer to simplifying release (GH442).
- Added doctr to automatically build docs for gh-pages (GH459).

2.5.7 v0.5.0 (July 25, 2017)

This is a major release from 0.4.0. We recommend that all users upgrade.

Highlights include:

- Compat with the new Yahoo iCharts API. Yahoo removed the older API, this release restores ability to download from Yahoo. (GH315)

What's new in v0.5.0

Enhancements

- `DataReader` now supports Quandl, see *here* (GH361).

Backwards incompatible API changes

- Removed Oanda as it became subscription only (GH296).

Bug Fixes

- web sessions are closed properly at the end of use (GH355)
- Handle commas in large price quotes (GH345)
- Test suite fixes for test_get_options_data (GH352)
- Test suite fixes for test_wdi_download (GH350)
- avoid monkey patching requests.Session (GH301)
- `get_data_yahoo()` now treats `'null'` strings as missing values (GH342)

2.5.8 v0.4.0 (May 15, 2017)

This is a major release from 0.3.0 and includes compat with pandas 0.20.1, and some backwards incompatible API changes.

Highlights include:

What's new in v0.4.0

- *Enhancements*
- *Backwards incompatible API changes*

Enhancements

- Compat with pandas 0.20.1 (GH304, GH320)
- Switched test framework to use `pytest` (GH310, GH312)

Backwards incompatible API changes

- Support has been dropped for Python 2.6 and 3.4 (GH313)
- Support has been dropped for *pandas* versions before 0.17.0 (GH313)

2.5.9 v0.3.0 (January 14, 2017)

This is a major release from 0.2.1 and includes new features and a number of bug fixes.

Highlights include:

What's new in v0.3.0

- *New features*

New features

- `DataReader` now supports dividend only pulls from Yahoo! Finance (GH138).
- `DataReader` now supports downloading mutual fund prices from the Thrift Savings Plan, see *here* (GH157).
- `DataReader` now supports Google options data source (GH148).
- `DataReader` now supports Google quotes (GH188).
- `DataReader` now supports Enigma dataset. see *here* (GH245).
- `DataReader` now supports downloading a full list of NASDAQ listed symbols. see *here* (GH254).

Other enhancements

- Eurostat reader now supports larger data returned from API via zip format. (GH205)
- Added support for Python 3.6.
- Added support for pandas 19.2

Bug Fixes

- Fixed bug that caused `DataReader` to fail if company name has a comma. (GH85).
- Fixed bug in `YahooOptions` caused as a result of change in yahoo website format. (GH244).

2.5.10 v0.2.1 (November 26, 2015)

This is a minor release from 0.2.0 and includes new features and bug fixes.

Highlights include:

What's new in v0.2.1

New features

- `DataReader` now supports Eurostat data sources, see *here* (GH101).
- `Options` downloading is approximately 4x faster as a result of a rewrite of the parsing function. (GH122)
- `DataReader` and `Options` now support caching, see *here* (GH110),(GH116),(GH121), (GH122).

Backwards incompatible API changes

- `Options` columns `PctChg` and `IV` (Implied Volatility) are now type float rather than string. (GH122)

2.5.11 v0.2.0 (October 9, 2015)

This is a major release from 0.1.1 and includes new features and a number of bug fixes.

Highlights include:

What's new in v0.2.0

New features

- Added latitude and longitude to output of wb.get_countries (GH47).
- Extended DataReader to fetch dividends and stock splits from Yahoo (GH45).
- Added get_available_datasets to famafrench (GH56).
- `DataReader` now supports OECD data sources, see *here* (GH101).

Backwards incompatible API changes

- Fama French indexes are not Pandas.PeriodIndex for annual and monthly data, and pandas.DatetimeIndex otherwise (GH56).

Bug Fixes

- Update Fama-French URL (GH53)
- Fixed bug where get_quote_yahoo would fail if a company name had a comma (GH85)

CHAPTER 3

Documentation

Stable documentation is available on github.io. A second copy of the stable documentation is hosted on read the docs for more details.

CHAPTER 4

Recent developments

You can install the latest development version using

```
pip install git+https://github.com/pydata/pandas-datareader.git
```

or

```
git clone https://github.com/pydata/pandas-datareader.git
cd pandas-datareader
python setup.py install
```

Development documentation is available for the latest changes in master.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

B

C

D

E

F

G

Q

R

S

T

U

W

Y