Up to date remote data access for pandas, works for multiple versions of pandas.

**Warning:** As of v0.6.0 Yahoo!, Google Options, Google Quotes and EDGAR have been immediately deprecated due to large changes in their API and no stable replacement.

**Note:** As of v0.6.0 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.
Starting in 0.19.0, pandas no longer supports `pandas.io.data` or `pandas.io.wb`, so you must replace your imports from `pandas.io` with those from `pandas_datareader`:

```python
from pandas.io import data, wb  # becomes
from pandas_datareader import data, wb
```

Many functions from the data module have been included in the top level API.

```python
import pandas_datareader as pdr
pdr.get_data_fred('GS10')
```
Stable documentation is available on github.io. A second copy of the stable documentation is hosted on read the docs for more details.

Development documentation is available for the latest changes in master.
3.1 Requirements

Using pandas datareader requires the following packages:

- pandas>=0.19.2
- lxml
- requests>=2.3.0
- requests-file
- requests-ftp
- wrapt

Building the documentation additionally requires:

- matplotlib
- ipython
- sphinx
- sphinx_rtd_theme

Testing requires pytest.

3.2 Install latest release version via pip

$ pip install pandas-datareader
3.3 Install latest development version

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

or

```bash
$ git clone https://github.com/pydata/pandas-datareader.git
$ python setup.py install
```
4.1 What’s New

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

4.1.1 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).

What’s new in v0.6.0

• **Enhancements**
  • Backwards incompatible API changes
  • Bug Fixes
  • Other Changes

**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)

**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).
• The end point for FRED was updated to the latest values (GH436).
• The 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).

4.1.2 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
- Backwards incompatible API changes
- Bug Fixes

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)

4.1.3 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)

4.1.4 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
  - Other enhancements
- Bug Fixes

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).

4.1.5 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
- Backwards incompatible API changes

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)

4.1.6 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
- Backwards incompatible API changes
- Bug Fixes

New features

- Added latitude and longitude to output of wb.get_countries (GH47).
- ExtendedDataReader 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)

4.2 Remote Data Access

**Warning:** Yahoo! Finance has been immediately deprecated. Yahoo! substantially altered their API in late 2017 and the csv endpoint was retired.

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:

- Google Finance
- Morningstar
- IEX
- Robinhood
- Enigma
- Quandl
- St.Louis FED (FRED)
- Kenneth French's data library
- World Bank
- OECD
- Eurostat
- Thrift Savings Plan
- Nasdaq Trader symbol definitions
- Stooq
- MOEX

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.
4.2.1 Google Finance

**Warning:** Google’s API has become less reliable during 2017. While the google datarreader often works as expected, it is not uncommon to experience a range of errors when attempting to read data, especially in bulk.

```python
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]: f = web.DataReader('F', 'google', start, end)
In [6]: f.ix['2010-01-04']
```

```
  →
  Open   10.17
  High   10.28
  Low    10.05
  Close  10.28
  Volume 60855796.00
Name: 2010-01-04 00:00:00, dtype: float64
```

**4.2.2 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).

```python
In [7]: import os
In [8]: import pandas_datareader as pdr
In [9]: df = pdr.get_data_tiingo('GOOG', api_key=os.getenv('TIINGO_API_KEY'))
```

```
Traceback (most recent call last)
  <ipython-input-9-c390cc1bcafe> in <module>()
    1 df = pdr.get_data_tiingo('GOOG', api_key=os.getenv('TIINGO_API_KEY'))
AttributeError: module 'pandas_datareader' has no attribute 'get_data_tiingo'
```

```python
In [10]: df.head()
```

```
Traceback (most recent call last)
  <ipython-input-10-c42a15b2c7cf> in <module>()
    1 df.head()
NameError: name 'df' is not defined
```

4.2. Remote Data Access
4.2.3 Morningstar

OHLC and Volume data is available from Morningstar using the same API which powers their charts.

```python
In [11]: import pandas_datareader.data as web
In [12]: from datetime import datetime
In [13]: start = datetime(2015, 2, 9)
In [14]: end = datetime(2017, 5, 24)
In [15]: f = web.DataReader('F', 'morningstar', start, end)
In [16]: f.head()
```

```
Out[16]:
         Close   High   Low   Open   Volume
Symbol Date
F  2015-02-09  15.92  16.03  15.72  15.76  20286720
    2015-02-10  16.09  16.14  15.91  16.05  27928530
    2015-02-11  16.25  16.31  16.01  16.08  34285331
    2015-02-12  16.36  16.45  16.30  16.34  23738806
    2015-02-13  16.30  16.36  16.19  16.33  19954568
```

4.2.4 IEX

The Investors Exchange (IEX) provides a wide range of data through an API. Historical stock prices are available for up to 5 years:

```python
In [17]: import pandas_datareader.data as web
In [18]: from datetime import datetime
In [19]: start = datetime(2015, 2, 9)
In [20]: end = datetime(2017, 5, 24)
In [21]: f = web.DataReader('F', 'iex', start, end)
5y
In [22]: f.loc['2015-02-09']
```

```
  open   high   low   close  volume
Name: 2015-02-09, dtype: float64
```

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Date</th>
<th>Close</th>
<th>High</th>
<th>Low</th>
<th>Open</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>2015-02-09</td>
<td>15.92</td>
<td>16.03</td>
<td>15.72</td>
<td>15.76</td>
<td>20286720</td>
</tr>
<tr>
<td></td>
<td>2015-02-10</td>
<td>16.09</td>
<td>16.14</td>
<td>15.91</td>
<td>16.05</td>
<td>27928530</td>
</tr>
<tr>
<td></td>
<td>2015-02-11</td>
<td>16.25</td>
<td>16.31</td>
<td>16.01</td>
<td>16.08</td>
<td>34285331</td>
</tr>
<tr>
<td></td>
<td>2015-02-12</td>
<td>16.36</td>
<td>16.45</td>
<td>16.30</td>
<td>16.34</td>
<td>23738806</td>
</tr>
<tr>
<td></td>
<td>2015-02-13</td>
<td>16.30</td>
<td>16.36</td>
<td>16.19</td>
<td>16.33</td>
<td>19954568</td>
</tr>
</tbody>
</table>

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

```python
In [23]: import pandas_datareader.data as web
In [24]: f = web.DataReader('gs', 'iex-tops')
In [25]: f[:10]
```
4.2.5 Robinhood

Robinhood is a stock trading platform with an API that provides a limited set of data. Historical daily data is limited to 1 year relative to today.

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

### In [27]:
```python
from datetime import datetime
```

### In [28]:
```python
f = web.DataReader('F', 'robinhood')
```

### Out[28]:

```py
Out[28]:
+-----------------+----------+-----------------+----------+
|                   | close_price | high_price      | interpolated |
| symbol | begins_at |      |      |       |
|--------|-----------|----------------|-------------|
| F      | 2017-02-02 | 11.5323 | 11.6169 | False |
|        | 2017-02-03 | 11.7953 | 11.8516 | False |
|        | 2017-02-06 | 11.7577 | 11.8469 | False |
|        | 2017-02-07 | 11.5887 | 11.7577 | False |
|        | 2017-02-08 | 11.6263 | 11.6920 | False |
```

### Out[29]:

```py
Out[29]:
+-----------------+----------+----------+----------+
|                   | session  |           | volume   |
| symbol | begins_at |      |      |       |
|--------|-----------|----------------|-------------|
| F      | 2017-02-02 | reg 29035383 |      |      |
|        | 2017-02-03 | reg 38245251 |      |      |
|        | 2017-02-06 | reg 26916768 |      |      |
|        | 2017-02-07 | reg 32914413 |      |      |
|        | 2017-02-08 | reg 26411417 |      |      |
```

4.2.6 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 [30]:
```python
import os
```

### In [31]:
```python
import pandas_datareader as pdr
```

### In [32]:
```python
df = pdr.get_data_enigma('292129b0-1275-44c8-a6a3-2a0881f24fe1', os.getenv('ENIGMA_API_KEY')
```

4.2. Remote Data Access
4.2.7 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 [34]: import pandas_datareader.data as web

In [35]: symbol = 'WIKI/AAPL'  # or 'AAPL.US'

In [36]: df = web.DataReader(symbol, 'quandl', '2015-01-01', '2015-01-05')

In [37]: df.loc['2015-01-02']
Out[37]:
### 4.2.8 FRED

```python
In [38]: import pandas_datareader.data as web
In [39]: import datetime
In [40]: start = datetime.datetime(2010, 1, 1)
In [41]: end = datetime.datetime(2013, 1, 27)
In [42]: gdp = web.DataReader('GDP', 'fred', start, end)
In [43]: gdp.ix['2013-01-01']
Out[43]:
GDP  16475.44
Name: 2013-01-01 00:00:00, dtype: float64

# Multiple series:
In [44]: inflation = web.DataReader(['CPIAUCSL', 'CPILFESL'], 'fred', start, end)
In [45]: inflation.head()
Out[45]:
     CPIAUCSL  CPILFESL
DATE
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
```

### 4.2.9 Fama/French

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

```python
In [46]: from pandas_datareader.famafrench import get_available_datasets
In [47]: import pandas_datareader.data as web
In [48]: len(get_available_datasets())
Out[48]: 262
```
In [49]: ds = web.DataReader('5_Industry_Portfolios', 'famafrench')

In [50]: print(ds['DESCR'])

5 Industry Portfolios

This file was created by CMPT_IND_RETgs using the 201712 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 2017 Kenneth R. French

0 : Average Value Weighted Returns -- Monthly (96 rows x 5 cols)
1 : Average Equal Weighted Returns -- Monthly (96 rows x 5 cols)
2 : Average Value Weighted Returns -- Annual (8 rows x 5 cols)
3 : Average Equal Weighted Returns -- Annual (8 rows x 5 cols)
4 : Number of Firms in Portfolios (96 rows x 5 cols)
5 : Average Firm Size (96 rows x 5 cols)
6 : Sum of BE / Sum of ME (8 rows x 5 cols)
7 : Value-Weighted Average of BE/ME (8 rows x 5 cols)

In [51]: ds[4].head()

4.2.10 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]: mathces = 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)
The resulting dataset is a properly formatted DataFrame with a hierarchical index, so it is easy to apply .groupby transformations to it:

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

```

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

```python
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 telephon... 
```

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

```python
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()
In [15]: dat.columns = ['gdp', 'cellphone']
In [16]: print(dat.tail())
```

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:

```python
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())
```

```
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.

### 4.2.11 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’.

```python
In [52]: import pandas_datareader.data as web
In [53]: import datetime
In [54]: df = web.DataReader('TUD', 'ocecd', end=datetime.datetime(2012, 1, 1))
In [55]: df.columns
Out[55]:
MultiIndex(levels=[['Australia', 'Austria', 'Belgium', 'Canada', 'Chile', 'Czech Republic', 'Denmark', 'Estonia', 'Finland', 'France', 'Germany', 'Greece', 'Hungary', 'Iceland', 'Ireland', 'Israel', 'Italy', 'Japan', 'Korea', 'Latvia', 'Lithuania', 'Luxembourg', 'Mexico', 'Netherlands', 'New Zealand', 'Norway', 'Poland', 'Portugal', 'Slovak Republic', 'Slovenia', 'Spain', 'Sweden', 'Switzerland', 'Turkey', 'United Kingdom', 'United States'], ['Annual', 'Administrative data', 'Survey data', 'Employees', 'Trade union density', 'Union members'], ['Percentage', 'Thousands']])
```

4.2. Remote Data Access
```python
In [56]: df[['Japan', 'United States']]
```

<table>
<thead>
<tr>
<th>Country</th>
<th>Frequency</th>
<th>Source</th>
<th>Series</th>
<th>Measure</th>
<th>Year</th>
<th>Country</th>
<th>Frequency</th>
<th>Source</th>
<th>Series</th>
<th>Measure</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan</td>
<td>Annual</td>
<td>Survey data</td>
<td>Union members</td>
<td>Trade union density</td>
<td>Employees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Japan</td>
<td>Annual</td>
<td>Survey data</td>
<td>Union members</td>
<td>Trade union density</td>
<td>Employees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Japan</td>
<td>Annual</td>
<td>Survey data</td>
<td>Union members</td>
<td>Trade union density</td>
<td>Employees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Frequency</th>
<th>Source</th>
<th>Series</th>
<th>Measure</th>
<th>Year</th>
<th>Country</th>
<th>Frequency</th>
<th>Source</th>
<th>Series</th>
<th>Measure</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan</td>
<td>Annual</td>
<td>Survey data</td>
<td>Union members</td>
<td>Trade union density</td>
<td>Employees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Japan</td>
<td>Annual</td>
<td>Survey data</td>
<td>Union members</td>
<td>Trade union density</td>
<td>Employees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Japan</td>
<td>Annual</td>
<td>Survey data</td>
<td>Union members</td>
<td>Trade union density</td>
<td>Employees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Frequency</th>
<th>Source</th>
<th>Series</th>
<th>Measure</th>
<th>Year</th>
<th>Country</th>
<th>Frequency</th>
<th>Source</th>
<th>Series</th>
<th>Measure</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan</td>
<td>Annual</td>
<td>Survey data</td>
<td>Union members</td>
<td>Trade union density</td>
<td>Employees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Japan</td>
<td>Annual</td>
<td>Survey data</td>
<td>Union members</td>
<td>Trade union density</td>
<td>Employees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Frequency</th>
<th>Source</th>
<th>Series</th>
<th>Measure</th>
<th>Year</th>
<th>Country</th>
<th>Frequency</th>
<th>Source</th>
<th>Series</th>
<th>Measure</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan</td>
<td>Annual</td>
<td>Survey data</td>
<td>Union members</td>
<td>Trade union density</td>
<td>Employees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Japan</td>
<td>Annual</td>
<td>Survey data</td>
<td>Union members</td>
<td>Trade union density</td>
<td>Employees</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[3 rows x 24 columns]
4.2.12 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:


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

In [57]: import pandas_datareader.data as web
In [58]: df = web.DataReader('tran_sf_railac', 'eurostat')
In [59]: df
Out[59]:

<table>
<thead>
<tr>
<th>ACCIDENT</th>
<th>UNIT</th>
<th>GEO</th>
<th>FREQ</th>
<th>TIME_PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2010-01-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2011-01-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2012-01-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2013-01-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2014-01-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2015-01-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2016-01-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7.0</td>
</tr>
</tbody>
</table>

ACCIDENT | UNIT   | GEO                  | FREQ | TIME_PERIOD |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Belgium Bulgaria</td>
<td>Annual</td>
<td>2010-01-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Switzerland Channel Tunnel Czech Republic</td>
<td>Annual</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Germany (until 1990 former territory of the FRG) Denmark Estonia</td>
<td>Annual</td>
<td>6.0</td>
</tr>
</tbody>
</table>
### 4.2.13 TSP Fund Data

Download mutual fund index prices for the TSP.

```python
In [60]: import pandas_datareader.tsp as tsp

In [61]: tspreader = tsp.TSPReader(start='2015-10-1', end='2015-12-31')

In [62]: tspreader.read()
```

```
Out[62]:
<table>
<thead>
<tr>
<th>date</th>
<th>L Income</th>
<th>L 2020</th>
<th>L 2030</th>
<th>L 2040</th>
<th>L 2050</th>
<th>G Fund</th>
<th>F Fund</th>
</tr>
</thead>
</table>
```

C Fund  S Fund  I Fund

<table>
<thead>
<tr>
<th>Date</th>
<th>C Fund</th>
<th>S Fund</th>
<th>I Fund</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-10-01</td>
<td>25.7953</td>
<td>34.0993</td>
<td>NaN</td>
</tr>
<tr>
<td>2015-10-02</td>
<td>26.1669</td>
<td>34.6504</td>
<td>23.6367</td>
</tr>
<tr>
<td>2015-10-05</td>
<td>26.6467</td>
<td>35.3565</td>
<td>24.1475</td>
</tr>
<tr>
<td>2015-10-06</td>
<td>26.5513</td>
<td>35.1320</td>
<td>24.2294</td>
</tr>
<tr>
<td>2015-10-07</td>
<td>26.7751</td>
<td>35.6035</td>
<td>24.3671</td>
</tr>
<tr>
<td>2015-10-08</td>
<td>27.0115</td>
<td>35.9016</td>
<td>24.6406</td>
</tr>
<tr>
<td>2015-10-09</td>
<td>27.0320</td>
<td>35.9772</td>
<td>24.7723</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2015-12-22</td>
<td>27.4848</td>
<td>35.0903</td>
<td>23.8679</td>
</tr>
<tr>
<td>2015-12-23</td>
<td>27.8272</td>
<td>35.5749</td>
<td>24.3623</td>
</tr>
<tr>
<td>2015-12-24</td>
<td>27.7831</td>
<td>35.6084</td>
<td>24.3272</td>
</tr>
<tr>
<td>2015-12-28</td>
<td>27.7230</td>
<td>35.4625</td>
<td>24.2816</td>
</tr>
<tr>
<td>2015-12-29</td>
<td>28.0236</td>
<td>35.8047</td>
<td>24.4757</td>
</tr>
<tr>
<td>2015-12-30</td>
<td>27.8239</td>
<td>35.5126</td>
<td>24.4184</td>
</tr>
<tr>
<td>2015-12-31</td>
<td>27.5622</td>
<td>35.2356</td>
<td>24.0952</td>
</tr>
</tbody>
</table>

[62 rows x 11 columns]

4.2.14 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.ix['IBM'])
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nasdaq Traded</td>
<td>True</td>
</tr>
<tr>
<td>Security Name</td>
<td>International Business Machines Corporation Co...</td>
</tr>
<tr>
<td>Listing Exchange</td>
<td>N</td>
</tr>
<tr>
<td>Market Category</td>
<td>ETF</td>
</tr>
<tr>
<td>ETF</td>
<td>True</td>
</tr>
<tr>
<td>Round Lot Size</td>
<td>100</td>
</tr>
<tr>
<td>Test Issue</td>
<td>False</td>
</tr>
<tr>
<td>Financial Status</td>
<td>NaN</td>
</tr>
<tr>
<td>CQS Symbol</td>
<td>IBM</td>
</tr>
<tr>
<td>NASDAQ Symbol</td>
<td>IBM</td>
</tr>
<tr>
<td>NextShares</td>
<td>False</td>
</tr>
<tr>
<td>Name</td>
<td>IBM, dtype: object</td>
</tr>
</tbody>
</table>

4.2.15 Stooq Index Data

Google finance doesn’t provide common index data download. The Stooq site has the data for download.
In [63]: import pandas_datareader.data as web

In [64]: f = web.DataReader('^DJI', 'stooq')

In [65]: f[:10]
Out[65]:
                   Open    High     Low    Close      Volume
Date
2018-02-01 26083.04 26306.70 26014.44 26186.71   NaN
2018-01-31 26268.17 26338.03 26050.98 26149.39 140120144.0
2018-01-30 26198.45 26256.99 26028.42 26076.89   11840144.0
2018-01-29 26584.28 26608.90 26435.34 26439.48  110919888.0
2018-01-26 26466.74 26616.71 26425.35 26616.71  123610888.0
2018-01-25 26313.06 26458.90 26259.72 26392.79  95732448.0
2018-01-24 26282.07 26392.80 26106.94 26252.12  123271104.0
2018-01-23 26214.87 26246.19 26143.90 26210.81 109272288.0
2018-01-22 26025.32 26215.23 25974.65 26214.60 126357768.0
2018-01-19 25987.35 26071.72 25942.83 26071.72 171541424.0

4.2.16 MOEX Data

The Moscow Exchange (MOEX) provides historical data.

In [66]: import pandas_datareader.data as web

In [67]: f = web.DataReader('USD000UTSTOM', 'moex', start='2017-07-01', end='2017-07-31')

In [68]: f.head()
Out[68]:
                   BOARDID SHORTNAME SECID  OPEN  LOW   HIGH  CLOSE  NUMTRADES VOLRUR WAPRICE
TRADEDATE
2017-07-03    CNGD    USDRUB_TOM  USD000UTSTOM  58.98  58.840  59.4250  59.3600     24   1.864785e+09      NaN
2017-07-04    CETS    USDRUB_TOM  USD000UTSTOM  59.30  59.135  59.4575  59.4125  21053  1.090265e+11  59.2700
2017-07-04    CNGD    USDRUB_TOM  USD000UTSTOM  59.36  59.830  59.3600  59.3575     37  1.046416e+09      NaN
2017-07-05    CNGD    USDRUB_TOM  USD000UTSTOM  59.30  59.300  60.2600  59.9825    50108  2.874226e+11  59.9234
2017-07-05    CNGD    USDRUB_TOM  USD000UTSTOM  59.34  59.265  60.1800  60.1800     35  6.339036e+09      NaN

4.3 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.

```python
In [1]: import pandas_datareader.data as web
In [2]: import datetime
In [3]: import requests_cache
In [4]: expire_after = datetime.timedelta(days=3)
In [5]: session = requests_cache.CachedSession(cache_name='cache', backend='sqlite',
   expire_after=expire_after)
In [6]: start = datetime.datetime(2010, 1, 1)
In [7]: end = datetime.datetime(2013, 1, 27)
In [8]: f = web.DataReader("F", 'yahoo', start, end, session=session)
---
```

```
ImmediateDeprecationError: Yahoo Daily has been immediately deprecated due to large breaks in the API without the introduction of a stable replacement. Pull Requests to re-enable these data connectors are welcome.

See https://github.com/pydata/pandas-datareader/issues
```

```python
In [9]: f.ix['2010-01-04']
```

```
NameError: name 'f' is not defined
```
4.4 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.

4.4.1 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.

4.4.2 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.

4.4.3 Barchart

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

A python package simplifying access is available on github.

4.4.4 List of Other Sources

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

4.5 Data Readers

4.5.1 Federal Reserve Economic Data (FRED)

```python
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

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 indices of each series.

Return type DataFrame
```
4.5.2 Fama-French Data (Ken French’s Data Library)

```python
class pandas_datareader.famafrench.FamaFrenchReader:
    def __init__(self, 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

        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
```

```python
url
API URL
```

```python
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.
```

4.5.3 Bank of Canada

```python
class pandas_datareader.bankofcanada.BankOfCanadaReader:
    def __init__(self, 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.
```

```python
url
API URL
```

```python
pandas_datareader.bankofcanada.get_available_datasets(**kwargs)
Get the list of datasets available from the Bank of Canada.

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

Returns

Return type A list of valid inputs for get_data_bankofcanada.
```
Notes

See Bank of Canada

`close()`
Close network session

`params`
Parameters to use in API calls

`read()`
Read data from connector

`url`
API URL

4.5.4 Engima

```python
class pandas_datareader.enigma.EnigmaReader (dataset_id=None, api_key=None, retry_count=5, pause=0.75, session=None)
```
Collects current snapshot of Enigma data located at the specified data set ID and returns a pandas DataFrame.

Examples

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

```python
>>> 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`

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

`close()`
Close network session

`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
4.5.5 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

dsd_url
API DSD URL

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

4.5.6 The Investors Exchange (IEX)

class pandas_datareader.iex.daily.IEXDailyReader(symbols=None, start=None, end=None, retry_count=3, pause=0.35, session=None, chunksize=25)

Returns DataFrame/Panel of historical stock prices from symbols, over date range, start to end. To avoid being penalized by Google 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, (defaults to '1/1/2010')) – Starting date, timestamp. Parses many different kind of date representations (e.g., ‘JAN-01-2010’, ‘1/1/10’, ‘Jan, 1, 1980’)

• end (string, (defaults to today)) – Ending date, timestamp. Same format as starting date.

• retry_count (int, default 3) – Number of times to retry query request.

• pause (int, default 0) – 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

close()
Close network session

endpoint
API endpoint
params
Parameters to use in API calls

read()
Read data

url
API URL

```python
class pandas_datareader.iex.market.MarketReader(symbols=None, start=None, end=None, retry_count=3, pause=0.001, 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.

```python
close()
Close network session
```

params
Parameters to use in API calls

read()
Read data

service
Service endpoint

url
API URL

```python
class pandas_datareader.iex.ref.SymbolsReader(symbols=None, start=None, end=None, retry_count=3, pause=0.001, session=None)
```

Symbols available for trading on IEX

**Notes**

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

```python
close()
Close network session
```

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=0.001, session=None)

Daily statistics from IEX for a day or month

close()
Close network session

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=0.001, session=None)

Monthly statistics from IEX

close()
Close network session

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=0.001, session=None)

Total matched volume information from IEX

close()
Close network session

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=0.001, 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

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=0.001, 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

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=0.001, 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

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=0.001, 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

params  
Parameters to use in API calls

read()  
Read data

service  
Service endpoint

url  
API URL
4.5.7 Moscow Exchange (MOEX)

class pandas_datareader.moex.MoexReader(*args, **kwargs)

Returns DataFrame of historical stock prices from symbols from Moex

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** (str, (defaults to '1/1/2010')) – Starting date, timestamp. Parses many different kind of date representations (e.g., ‘JAN-01-2010’, ‘1/1/10’, ‘Jan, 1, 1980’)
- **end** (str, (defaults to today)) – Ending date, timestamp. Same format as starting date.
- **retry_count** (int, default 3) – Number of times to retry query request.
- **pause** (int, default 0) – 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

Notes

To avoid being penalized by Moex servers, pauses between downloading ‘chunks’ of symbols can be specified.

Methods

- **close()**
  Close network session

- **params**
  Parameters to use in API calls

- **read()**
  Read data

- **url**
  API URL

4.5.8 Morningstar

class pandas_datareader.mstar.daily.MorningstarDailyReader(symbols, start=None, end=None, retry_count=3, pause=0.1, timeout=30, session=None, freq=None, incl_splits=False, incl_dividends=False, incl_volume=True, currency='usd', interval='d')

Read daily data from Morningstar
Parameters

- `symbols` (*str, List[str]*) – String symbol of like of symbols
- `start` (*string, (defaults to '1/1/2010')) – Starting date, timestamp. Parses many different kind of date representations (e.g., ‘JAN-01-2010’, ‘1/1/10’, ‘Jan, 1, 1980’)
- `end` (*string, (defaults to today)) – 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) – Frequency to use in select readers
- `incl_splits` (*bool, optional) – Include splits in data
- `incl_dividends` (*bool, optional) – Include dividends in data
- `incl_volume` (*bool, optional) – Include volume in data
- `currency` (*str, optional) – Currency to use for data
- `interval` (*str, optional) – Sampling interval to use for downloaded data

Notes

See Morningstar

```python
close()
Close network session

params
Parameters to use in API calls

read()
Read data

url
API URL
```

4.5.9 NASDAQ

```python
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

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

```python
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.

4.5. Data Readers
close()
    Close network session

params
    Parameters to use in API calls

read()
    Read data from connector

url
    API URL

4.5.11 Quandl

class pandas_datareader.quandl.QuandlReader(symbols=None, start=None, end=None, retry_count=3, pause=0.001, session=None, chunksize=25)

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) – Starting date, timestamp. Parses many different kind of date representations (e.g., ‘JAN-01-2010’, ‘1/1/10’, ‘Jan, 1, 1980’)

* end (string, (defaults to today)) – Ending date, timestamp. Same format as starting date.

* retry_count (int, default 3) – Number of times to retry query request.

* pause (int, default 0) – 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
4.5.12 Robinhood

class pandas_datareader.robinhood.RobinhoodHistoricalReader(symbols, start=None, end=None, retry_count=3, pause=0.1, timeout=30, session=None, freq=None, interval='day', span='year'))

Read historical values from Robinhood

Parameters

- symbols ([str, List[str]]) – String symbol of like of symbols
- start (None) – Ignored. See span and interval.
- end (None) – Ignored. See span and interval.
- 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 (None) – Quotes are near real-time and so this value is ignored
- interval (["day", "week", "5minute", "10minute"]) – Interval between historical prices
- span (["day", "week", "year", "5year"]) – Time span relative to now to retrieve. The available spans are a function of interval. See notes

Notes

Only provides up to 1 year of daily data.

The available spans are a function of interval.

- day: year
- week: 5year
- 5minute: day, week
- 10minute: day, week

close()
Close network session

params
Parameters to use in API calls

read()
Read data from connector

url
API URL
class pandas_datareader.robinhood.RobinhoodQuoteReader(symbols, start=None, end=None, retry_count=3, pause=0.1, timeout=30, session=None, freq=None)

Read quotes from Robinhood

Parameters
- **symbols** (str, List[str]) – String symbol of like of symbols
- **start** (None) – Quotes are near real-time and so this value is ignored
- **end** (None) – Quotes are near real-time and so this value is ignored
- **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** (None) – Quotes are near real-time and so this value is ignored

close()
Close network session

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

4.5.13 Stooq.com

class pandas_datareader.stooq.StooqDailyReader(symbols=None, start=None, end=None, retry_count=3, pause=0.001, session=None, chunksize=25)

Returns DataFrame/Panel of historical stock prices from symbols, over date range, start to end. To avoid being penalized by Google 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.
- **retry_count** (int, default 3) – Number of times to retry query request.
- **pause** (int, default 0) – 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

Notes

See Stooq
4.5.14 Tiingo

```python
class pandas_datareader.tiingo.TiingoDailyReader:
symbols, start=func, end=func,
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 of like of symbols
- `start` (str, (defaults to '1/1/2010')) – Starting date, timestamp. Parses many different kind of date representations (e.g., ‘JAN-01-2010’, ‘1/1/10’, ‘Jan, 1, 1980’)
- `end` (str, (defaults to today)) – 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.

```python
close()
Close network session
```

**params**

Parameters to use in API calls

```python
read()
Read data
```

**url**

API URL

```python
class pandas_datareader.tiingo.TiingoQuoteReader:
symbols, start=func, end=func,
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 of like of symbols
• **start**(*str*, *(defaults to '1/1/2010')) – Not used.
• **end**(*str*, *(defaults to today)) – 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.

```python
class pandas_datareader.tiingo.TiingoMetaDataReader(symbols=None, 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 of like of symbols
• **start**(*str*, *(defaults to '1/1/2010')*) – Not used.
• **end**(*str*, *(defaults to today)*) – 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**.

```python
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

4.5.15 Thrift Savings Plan (TSP)

class pandas_datareader.tsp.TSPReader(symbols=('Linc', 'L2020', 'L2030', 'L2040', 'L2050', 'G', 'F', 'C', 'S', 'I'), start=None, end=None, retry_count=3, pause=0.001, 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 (str, (defaults to '1/1/2010')) – Starting date, timestamp. Parses many different kind of date representations (e.g., ‘JAN-01-2010’, ‘1/1/10’, ‘Jan, 1, 1980’)

• end (str, (defaults to today)) – Ending date, timestamp. Same format as starting date.

• retry_count (int, default 3) – Number of times to retry query request.

• pause (int, default 0) – 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

4.5.16 World Bank

class pandas_datareader.wb.WorldBankReader(symbols=None, countries=None, start=None, end=None, freq=None, retry_count=3, pause=0.001, 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** *(Timestamp or int)* – First year of the data series. Month and day are ignored.

• **end** *(Timestamp or int)* – 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

**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.*cap', 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 col-
lection 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’) – Coun-
try codes are validated against a hardcoded list. This controls the outcome of that valida-
tion, 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, iso_code, 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.*capit’, 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**: 
CHAPTER 5

Indices and tables

- genindex
- modindex
- search
pandas_datareader.bankofcanada, 31
pandas_datareader.enigma, 32
pandas_datareader.eurostat, 33
pandas_datareader.fama_french, 31
pandas_datareader.fred, 30
pandas_datareader.iex.daily, 33
pandas_datareader.iex.deep, 36
pandas_datareader.iex.market, 34
pandas_datareader.iex.ref, 34
pandas_datareader.iex.stats, 34
pandas_datareader.iex.tops, 37
pandas_datareader.moex, 38
pandas_datareader.mstar.daily, 38
pandas_datareader.nasdaq_trader, 39
pandas_datareader.oecd, 39
pandas_datareader.quandl, 40
pandas_datareader.robinhood, 41
pandas_datareader.stooq, 42
pandas_datareader.tiingo, 43
pandas_datareader.tsp, 45
pandas_datareader.wb, 45
Index

B
BankOfCanadaReader (class in pandas_datareader.bankofcanada), 31
C
close() (pandas_datareader.bankofcanada.BankOfCanadaReader method), 32
close() (pandas_datareader.enigma.EnigmaReader method), 32
close() (pandas_datareader.eurostat.EurostatReader method), 33
close() (pandas_datareader.famafrench.FamaFrenchReader method), 31
close() (pandas_datareader.fred.FredReader method), 30
close() (pandas_datareader.iex.daily.IEXDailyReader method), 33
close() (pandas_datareader.iex.deep.Deep method), 36
close() (pandas_datareader.iex.market.MarketReader method), 34
close() (pandas_datareader.iex.ref.SymbolsReader method), 34
close() (pandas_datareader.iex.stats.DailySummaryReader method), 35
close() (pandas_datareader.iex.stats.MonthlySummaryReader method), 35
close() (pandas_datareader.iex.stats.RecentReader method), 36
close() (pandas_datareader.iex.stats.RecordsReader method), 35
close() (pandas_datareader.iex.tops.LastReader method), 37
close() (pandas_datareader.iex.tops.TopsReader method), 37
close() (pandas_datareader.moex.MoexReader method), 38
close() (pandas_datareader.mstar.daily.MorningstarDailyReader method), 39
close() (pandas_datareader.oecd.OECDReader method), 39
close() (pandas_datareader.quandl.QuandlReader method), 40
close() (pandas_datareader.robinhood.RobinhoodHistoricalReader method), 41
close() (pandas_datareader.robinhood.RobinhoodQuoteReader method), 42
close() (pandas_datareader.stats.tiingo.TiingoDailyReader method), 43
close() (pandas_datareader.stats.tiingo.TiingoMetaDataReader method), 44
close() (pandas_datareader.stats.tiingo.TiingoQuoteReader method), 44
close() (pandas_datareader.tsp.TSPReader method), 45
close() (pandas_datareader.wb.WorldBankReader method), 46
D
DailySummaryReader (class in pandas_datareader.iex.stats), 34
Deep (class in pandas_datareader.iex.deep), 36
download() (in module pandas_datareader.wb), 47
dsd_url (pandas_datareader.eurostat.EurostatReader attribute), 33
E
endpoint (pandas_datareader.iex.daily.IEXDailyReader attribute), 33
EnigmaReader (class in pandas_datareader.enigma), 32
EurostatReader (class in pandas_datareader.eurostat), 33
F
FamaFrenchReader (class in pandas_datareader.famafrench), 31
FredReader (class in pandas_datareader.fred), 30
G
get_available_datasets() (in module pandas_datareader.famafrench), 31
<table>
<thead>
<tr>
<th>Method/Property</th>
<th>Module/Class/Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>pandas_datareader.get_available_datasets()</td>
<td>pandas_datareader.famafrench.FamaFrenchReader method</td>
</tr>
<tr>
<td>pandas_datareader.get_countries()</td>
<td>pandas_datareader.wb.WorldBankReader method</td>
</tr>
<tr>
<td>pandas_datareader.get_current_snapshot_id()</td>
<td>pandas_datareader.enigma.EnigmaReader method</td>
</tr>
<tr>
<td>pandas_datareader.get_dataset_metadata()</td>
<td>pandas_datareader.enigma.EnigmaReader method</td>
</tr>
<tr>
<td>pandas_datareader.get_indicators()</td>
<td>pandas_datareader.wb.WorldBankReader method</td>
</tr>
<tr>
<td>pandas_datareader.get_nasdaq_symbols()</td>
<td>pandas_datareader.nasdaq_trader module</td>
</tr>
<tr>
<td>pandas_datareader.get_snapshot_export()</td>
<td>pandas_datareader.enigma.EnigmaReader method</td>
</tr>
<tr>
<td>pandas_datareader.get_tiingo_symbols()</td>
<td>pandas_datareader.tiingo module</td>
</tr>
<tr>
<td>IEXDailyReader</td>
<td>pandas_datareader.iex.daily module</td>
</tr>
<tr>
<td>LastReader</td>
<td>pandas_datareader.iex.tops module</td>
</tr>
<tr>
<td>MarketReader</td>
<td>pandas_datareader.iex.market module</td>
</tr>
<tr>
<td>MoexReader</td>
<td>pandas_datareader.moex module</td>
</tr>
<tr>
<td>MonthlySummaryReader</td>
<td>pandas_datareader.iex.stats module</td>
</tr>
<tr>
<td>MorningstarDailyReader</td>
<td>pandas_datareader.mstar.daily module</td>
</tr>
<tr>
<td>OECDReader</td>
<td>pandas_datareader.oecd module</td>
</tr>
<tr>
<td>pandas_datareader.bankofcanada</td>
<td>pandas_datareader.bankofcanada module</td>
</tr>
<tr>
<td>pandas_datareader.mstar.daily</td>
<td>pandas_datareader.mstar.daily module</td>
</tr>
<tr>
<td>pandas_datareader.nasdaq_trader</td>
<td>pandas_datareader.nasdaq_trader module</td>
</tr>
<tr>
<td>pandas_datareader.oecd</td>
<td>pandas_datareader.oecd module</td>
</tr>
<tr>
<td>pandas_datareader.robinhood</td>
<td>pandas_datareader.robinhood module</td>
</tr>
<tr>
<td>pandas_datareader.stooq</td>
<td>pandas_datareader.stooq module</td>
</tr>
<tr>
<td>pandas_datareader.tiingo</td>
<td>pandas_datareader.tiingo module</td>
</tr>
<tr>
<td>pandas_datareader.tsp</td>
<td>pandas_datareader.tsp module</td>
</tr>
<tr>
<td>pandas_datareader.wb</td>
<td>pandas_datareader.wb module</td>
</tr>
<tr>
<td>params (pandas_datareader.bankofcanada.BankOfCanadaReader attribute)</td>
<td>pandas_datareader.bankofcanada module</td>
</tr>
<tr>
<td>params (pandas_datareader.enigma.EnigmaReader attribute)</td>
<td>pandas_datareader.enigma module</td>
</tr>
<tr>
<td>params (pandas_datareader.eurostat.EurostatReader attribute)</td>
<td>pandas_datareader.eurostat module</td>
</tr>
<tr>
<td>params (pandas_datareader.famafrench.FamaFrenchReader attribute)</td>
<td>pandas_datareader.famafrench module</td>
</tr>
<tr>
<td>params (pandas_datareader.fred.FredReader attribute)</td>
<td>pandas_datareader.fred module</td>
</tr>
<tr>
<td>params (pandas_datareader.iex.daily.IEXDailyReader attribute)</td>
<td>pandas_datareader.iex.daily module</td>
</tr>
<tr>
<td>params (pandas_datareader.iex.deep.Deep attribute)</td>
<td>pandas_datareader.iex.deep module</td>
</tr>
<tr>
<td>params (pandas_datareader.iex.market.MarketReader attribute)</td>
<td>pandas_datareader.iex.market module</td>
</tr>
<tr>
<td>params (pandas_datareader.iex.ref.SymbolsReader attribute)</td>
<td>pandas_datareader.iex.ref module</td>
</tr>
<tr>
<td>params (pandas_datareader.iex.stats.DailySummaryReader attribute)</td>
<td>pandas_datareader.iex.stats module</td>
</tr>
<tr>
<td>params (pandas_datareader.iex.stats.MonthlySummaryReader attribute)</td>
<td>pandas_datareader.iex.stats module</td>
</tr>
<tr>
<td>params (pandas_datareader.iex.stats.RecentReader attribute)</td>
<td>pandas_datareader.iex.stats module</td>
</tr>
<tr>
<td>params (pandas_datareader.iex.stats.RecordsReader attribute)</td>
<td>pandas_datareader.iex.stats module</td>
</tr>
<tr>
<td>params (pandas_datareader.iex.tops.LastReader attribute)</td>
<td>pandas_datareader.iex.tops module</td>
</tr>
<tr>
<td>params (pandas_datareader.iex.tops.TopsReader attribute)</td>
<td>pandas_datareader.iex.tops module</td>
</tr>
<tr>
<td>params (pandas_datareader.moex.MoexReader attribute)</td>
<td>pandas_datareader.moex module</td>
</tr>
<tr>
<td>params (pandas_datareader.mstar.daily.MorningstarDailyReader attribute)</td>
<td>pandas_datareader.mstar.daily module</td>
</tr>
<tr>
<td>params (pandas_datareader.oecd.OECDReader attribute)</td>
<td>pandas_datareader.oecd module</td>
</tr>
<tr>
<td>params (pandas_datareader.robinhood.RobinhoodHistoricalReader attribute)</td>
<td>pandas_datareader.robinhood module</td>
</tr>
<tr>
<td>params (pandas_datareader.robinhood.RobinhoodQuoteReader attribute)</td>
<td>pandas_datareader.robinhood module</td>
</tr>
<tr>
<td>params (pandas_datareader.stooq.StooqDailyReader attribute)</td>
<td>pandas_datareader.stooq module</td>
</tr>
<tr>
<td>params (pandas_datareader.tiingo.TiingoDailyReader attribute)</td>
<td>pandas_datareader.tiingo module</td>
</tr>
</tbody>
</table>
params (pandas_datareader.tsp.TSPReader attribute), 45
params (pandas_datareader.wb.WorldBankReader attribute), 46

QuandlReader (class in pandas_datareader.quandl), 40

read() (pandas_datareader.bankofcanada.BankOfCanadaReader method), 32
read() (pandas_datareader.enigma.EnigmaReader method), 32
read() (pandas_datareader.eurostat.EurostatReader method), 33
read() (pandas_datareader.famafrench.FamaFrenchReader method), 31
read() (pandas_datareader.fred.FredReader method), 30
read() (pandas_datareader.iex.daily.IEXDailyReader method), 34
read() (pandas_datareader.iex.deep.Deep method), 36
read() (pandas_datareader.iex.market.MarketReader method), 34
read() (pandas_datareader.iex.ref.SymbolsReader method), 34
read() (pandas_datareader.iex.stats.DailySummaryReader method), 35
read() (pandas_datareader.iex.stats.MonthlySummaryReader method), 35
read() (pandas_datareader.iex.stats.RecordsReader method), 35
read() (pandas_datareader.iex.tops.LastReader method), 37
read() (pandas_datareader.iex.tops.TopsReader method), 37
read() (pandas_datareader.moex.MoexReader method), 38
read() (pandas_datareader.mstar.daily.MorningstarDailyReader method), 39
read() (pandas_datareader.oecd.OECDReader method), 40
read() (pandas_datareader.quandl.QuandlReader method), 40
read() (pandas_datareader.robinhood.RobinhoodReader method), 41
read() (pandas_datareader.robinhood.RobinhoodHistoricalReader method), 41
read() (pandas_datareader.robinhood.RobinhoodQuoteReader method), 41
read() (pandas_datareader.tiingo.TiingoDailyReader method), 43
read() (pandas_datareader.tiingo.TiingoMetaDataReader method), 44
read() (pandas_datareader.tiingo.TiingoQuoteReader method), 44
read() (pandas_datareader.tsp.TSPReader method), 45
read() (pandas_datareader.wb.WorldBankReader method), 46
RecentReader (class in pandas_datareader.iex.stats), 36
RecordsReader (class in pandas_datareader.iex.stats), 35
RobinhoodHistoricalReader (class in pandas_datareader.robinhood), 41
RobinhoodQuoteReader (class in pandas_datareader.robinhood), 41
search() (in module pandas_datareader.wb), 47
search() (pandas_datareader.wb.WorldBankReader method), 46
service (pandas_datareader.iex.deep.Deep attribute), 37
service (pandas_datareader.iex.market.MarketReader attribute), 34
service (pandas_datareader.iex.ref.SymbolsReader attribute), 34
service (pandas_datareader.iex.stats.DailySummaryReader attribute), 35
service (pandas_datareader.iex.stats.MonthlySummaryReader attribute), 35
service (pandas_datareader.iex.stats.RecordsReader attribute), 36
service (pandas_datareader.iex.stats.RecentReader attribute), 36
service (pandas_datareader.iex.stats.RecordsReader attribute), 35
service (pandas_datareader.iex.tops.LastReader attribute), 37
service (pandas_datareader.iex.tops.TopsReader attribute), 37
StooqDailyReader (class in pandas_datareader.stooq), 42
SymbolsReader (class in pandas_datareader.iex.ref), 34
TiingoDailyReader (class in pandas_datareader.tiingo), 43
TiingoMetaDataReader (class in pandas_datareader.tiingo), 44
TiingoQuoteReader (class in pandas_datareader.tiingo), 44
TopsReader (class in pandas_datareader.iex.tops), 37

Index
WorldBankReader (class in pandas_datareader.wb), 45