pandas-datareader Documentation

Release 0.6.0

pydata

Feb 02, 2018

Contents

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.

CHAPTER 1

Usage

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:

```
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.

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

CHAPTER 2

Documentation

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.

CHAPTER 3

Installation

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

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

or

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

CHAPTER 4

Documentation

Contents:

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

- 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).
- 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).

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

- `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

- 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

- `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).
- 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)

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'a API has become less reliable during 2017. While the google datareader often works as expected, it is not uncommon to experience a range of errors when attempting to read data, especially in bulk.

```
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).

```
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'))
---------------------------------------------------------------------------
AttributeError                            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'

In [10]: df.head()
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
↪---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-c42a15b2c7cf> in <module>()
----> 1 df.head()

NameError: name 'df' is not defined
```

4.2.3 Morningstar

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

```
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:

```
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']
\\\Out[22]:
open              15.76
high              16.03
low               15.72
close             15.92
volume      20286720.00
Name: 2015-02-09, dtype: float64
```

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.

```
In [23]: import pandas_datareader.data as web

In [24]: f = web.DataReader('gs', 'iex-tops')

In [25]: f[:10]
```

```
Out[25]:
                                   0
askPrice                           0
askSize                            0
bidPrice                           0
bidSize                            0
lastSalePrice                 272.48
lastSaleSize                       3
lastSaleTime           1517518796044
lastUpdated            1517518800000
marketPercent                0.02056
sector          diversifiedfinancials
```

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]: import pandas_datareader.data as web

In [27]: from datetime import datetime

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

In [29]: f.head()
Out[29]:
                  close_price high_price  interpolated low_price open_price  \
symbol begins_at
F      2017-02-02     11.5323    11.6169         False   11.4854    11.5511
       2017-02-03     11.7953    11.8516         False   11.6356    11.6638
       2017-02-06     11.7577    11.8469         False   11.7014    11.7859
       2017-02-07     11.5887    11.7577         False   11.5605    11.7389
       2017-02-08     11.6263    11.6920         False   11.5230    11.5887

                  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]: import os

In [31]: import pandas_datareader as pdr

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-32-f46ac2b42095> in <module>()
----> 1 df = pdr.get_data_enigma('292129b0-1275-44c8-a6a3-2a0881f24fe1', os.getenv(
↪'ENIGMA_API_KEY'))

~/checkouts/readthedocs.org/user_builds/pandas-datareader/envs/stable/lib/python3.5/
↪site-packages/pandas_datareader-0.6.0-py3.5.egg/pandas_datareader/data.py in get_
↪data_enigma(*args, **kwargs)
     66
     67 def get_data_enigma(*args, **kwargs):
---> 68     return EnigmaReader(*args, **kwargs).read()
     69
     70

~/checkouts/readthedocs.org/user_builds/pandas-datareader/envs/stable/lib/python3.5/
↪site-packages/pandas_datareader-0.6.0-py3.5.egg/pandas_datareader/enigma.py in __
↪init__(self, dataset_id, api_key, retry_count, pause, session)
     41             self._api_key = os.getenv('ENIGMA_API_KEY')
     42             if self._api_key is None:
---> 43                 raise ValueError("Please provide an Enigma API key or set "
     44                                  "the ENIGMA_API_KEY environment variable\n"
     45                                  "If you do not have an API key, you can get "

ValueError: Please provide an Enigma API key or set the ENIGMA_API_KEY environment␣
↪variable
If you do not have an API key, you can get one here: http://public.enigma.com/signup

In [33]: df.columns
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
↪---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-33-b666bf274d0a> in <module>()
----> 1 df.columns

NameError: name 'df' is not defined
```

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]:
```

```
              Open    High     Low   Close      Volume  ExDividend  \
Date
2015-01-02  111.39  111.44  107.35  109.33  53204626.0         0.0

            SplitRatio     AdjOpen     AdjHigh      AdjLow    AdjClose  \
Date
2015-01-02         1.0  105.820966  105.868466  101.982949  103.863957

             AdjVolume
Date
2015-01-02  53204626.0
```

4.2.8 FRED

```
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.

```
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_RETS 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()
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
↪
         Cnsmr  Manuf  HiTec  Hlth   Other
Date
2010-01    622    737    830    467   1232
2010-02    620    734    821    464   1221
2010-03    614    729    818    458   1215
2010-04    614    726    807    458   1203
2010-05    611    723    804    457   1195
```

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)
                      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()
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.

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'.

```
In [52]: import pandas_datareader.data as web

In [53]: import datetime

In [54]: df = web.DataReader('TUD', 'oecd', 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']],
           labels=[[13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 21, 21, 21, 21,␣
↪21, 21, 21, 21, 21, 21, 21, 21, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 18, 18, 18, 18,␣
↪18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 7,␣
↪7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 11,
↪ 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16,
↪ 16, 16, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 9, 9, 9, 9, 9, 9, 9, 9, 9,␣
↪9, 9, 9, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 3, 3, 3, 3, 3, 3, 3, 3, 3,␣
↪3, 3, 3, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 32, 32, 32, 32, 32, 32, 32,
↪ 32, 32, 32, 32, 32, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 10, 10, 10, 10,
↪ 10, 10, 10, 10, 10, 10, 10, 10, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 31, 31, 31, 31,
↪ 31, 31, 31, 31, 31, 31, 31, 31, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 34,
↪ 34, 34, 34, 34, 34, 34, 34, 34, 34, 34, 34, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 8,␣
↪8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 12,
↪ 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
↪ 25, 25, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 6, 6, 6, 6, 6, 6, 6, 6, 6,␣
↪6, 6, 6, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 4, 4, 4, 4, 4, 4, 4, 4, 4,␣
↪4, 4, 4, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 27, 27, 27, 27, 27, 27, 27,
↪ 27, 27, 27, 27, 27, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 26, 26, 26, 26,
↪ 26, 26, 26, 26, 26, 26, 26, 26, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 20, 20, 20, 20,
↪ 20, 20, 20, 20, 20, 20, 20, 20], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1, 1, 1,␣
↪1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0,␣
↪0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,␣
```

```
           names=['Country', 'Frequency', 'Source', 'Series', 'Measure'])

In [56]: df[['Japan', 'United States']]
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
Country            Japan                                                      \
Frequency         Annual
Source       Survey data
Series     Union members            Trade union  density            Employees
Measure        Thousands Percentage            Thousands Percentage Thousands
Year
2010-01-01           NaN        NaN                  NaN        NaN       NaN
2011-01-01           NaN        NaN                  NaN        NaN       NaN
2012-01-01           NaN        NaN                  NaN        NaN       NaN

Country                                                                   \
Frequency
Source                Administrative data
Series                      Union members            Trade union  density
Measure    Percentage           Thousands Percentage            Thousands
Year
2010-01-01        NaN             12417.5        NaN                  NaN
2011-01-01        NaN             12271.9        NaN                  NaN
2012-01-01        NaN             12227.1        NaN                  NaN

Country                   ...            United States                      \
Frequency                 ...                   Annual
Source                    ...              Survey data
Series                    ...      Trade union  density            Employees
Measure    Percentage     ...                Thousands Percentage Thousands
Year                      ...
2010-01-01       28.9     ...                      NaN       17.4   97406.0
2011-01-01       27.6     ...                      NaN       16.5  102403.0
2012-01-01       25.9     ...                      NaN       15.9  106924.0

Country                                                                   \
Frequency
Source                Administrative data
Series                      Union members            Trade union  density
Measure    Percentage           Thousands Percentage            Thousands
Year
2010-01-01        NaN                 NaN        NaN                  NaN
2011-01-01        NaN                 NaN        NaN                  NaN
2012-01-01        NaN                 NaN        NaN                  NaN

Country
Frequency
Source
Series                Employees
Measure    Percentage Thousands Percentage
Year
2010-01-01        NaN   97406.0        NaN
2011-01-01        NaN  102403.0        NaN
2012-01-01        NaN  106924.0        NaN

[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:

- 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 [57]: import pandas_datareader.data as web

In [58]: df = web.DataReader('tran_sf_railac', 'eurostat')

In [59]: df
Out[59]:
ACCIDENT    Collisions of trains, including collisions with obstacles within the␣
↪clearance gauge  \
UNIT                                                                            ␣
↪    Number
GEO                                                                             ␣
↪   Austria
FREQ                                                                            ␣
↪    Annual
TIME_PERIOD                                                                     ␣
↪
2010-01-01                                                     3.0              ␣
↪
2011-01-01                                                     2.0              ␣
↪
2012-01-01                                                     1.0              ␣
↪
2013-01-01                                                     4.0              ␣
↪
2014-01-01                                                     1.0              ␣
↪
2015-01-01                                                     7.0              ␣
↪
2016-01-01                                                     7.0              ␣
↪

ACCIDENT                                                             \
UNIT
GEO         Belgium Bulgaria Switzerland Channel Tunnel Czech Republic
FREQ         Annual   Annual      Annual         Annual         Annual
TIME_PERIOD
2010-01-01      5.0      2.0         5.0            0.0            3.0
2011-01-01      0.0      0.0         4.0            0.0            6.0
2012-01-01      3.0      3.0         4.0            0.0            6.0
2013-01-01      1.0      2.0         6.0            0.0            5.0
2014-01-01      3.0      4.0         0.0            0.0           13.0
2015-01-01      0.0      3.0         3.0            0.0           14.0
2016-01-01      2.0      3.0         2.0            0.0            6.0

ACCIDENT                                                                   \
UNIT
GEO         Germany (until 1990 former territory of the FRG) Denmark Estonia
FREQ                                                  Annual  Annual  Annual
```

```
TIME_PERIOD
2010-01-01                                                   13.0      0.0      1.0
2011-01-01                                                   18.0      1.0      0.0
2012-01-01                                                   23.0      1.0      3.0
2013-01-01                                                   29.0      0.0      0.0
2014-01-01                                                   32.0      0.0      0.0
2015-01-01                                                   40.0      3.0      0.0
2016-01-01                                                   29.0      0.0      3.0

ACCIDENT                 ...          Unknown                                  \
UNIT                     ...           Number
GEO          Greece      ...      Netherlands Norway Poland Portugal Romania
FREQ         Annual      ...           Annual Annual Annual   Annual  Annual
TIME_PERIOD              ...
2010-01-01      4.0      ...              NaN    NaN    NaN      NaN     NaN
2011-01-01      1.0      ...              NaN    NaN    NaN      NaN     NaN
2012-01-01      2.0      ...              NaN    NaN    NaN      NaN     NaN
2013-01-01      2.0      ...              NaN    NaN    NaN      NaN     NaN
2014-01-01      1.0      ...              NaN    NaN    NaN      NaN     NaN
2015-01-01      2.0      ...              NaN    NaN    NaN      NaN     NaN
2016-01-01      1.0      ...              NaN    NaN    NaN      NaN     NaN

ACCIDENT
UNIT
GEO          Sweden Slovenia Slovakia Turkey United Kingdom
FREQ         Annual   Annual   Annual Annual         Annual
TIME_PERIOD
2010-01-01      NaN      NaN      NaN    0.0            NaN
2011-01-01      NaN      NaN      NaN    0.0            NaN
2012-01-01      NaN      NaN      NaN    0.0            NaN
2013-01-01      NaN      NaN      NaN    0.0            NaN
2014-01-01      NaN      NaN      NaN    0.0            NaN
2015-01-01      NaN      NaN      NaN    0.0            NaN
2016-01-01      NaN      NaN      NaN    0.0            NaN

[7 rows x 264 columns]
```

4.2.13 TSP Fund Data

Download mutual fund index prices for the TSP.

```
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]:
             L Income   L 2020   L 2030   L 2040   L 2050   G Fund   F Fund  \
date
2015-10-01    17.5164  22.5789  24.2159  25.5690  14.4009  14.8380  17.0467
2015-10-02    17.5707  22.7413  24.4472  25.8518  14.5805  14.8388  17.0924
2015-10-05    17.6395  22.9582  24.7571  26.2306  14.8233  14.8413  17.0531
2015-10-06    17.6338  22.9390  24.7268  26.1898  14.7979  14.8421  17.0790
2015-10-07    17.6639  23.0324  24.8629  26.3598  14.9063  14.8429  17.0725
2015-10-08    17.6957  23.1364  25.0122  26.5422  15.0240  14.8437  17.0363
2015-10-09    17.7048  23.1646  25.0521  26.5903  15.0554  14.8445  17.0511
```

```
...               ...       ...       ...       ...       ...       ...       ...
2015-12-22   17.7493  23.1452  24.9775  26.4695  14.9611  14.9076  16.9607
2015-12-23   17.8015  23.3149  25.2208  26.7663  15.1527  14.9084  16.9421
2015-12-24   17.7991  23.3039  25.2052  26.7481  15.1407  14.9093  16.9596
2015-12-28   17.7950  23.2811  25.1691  26.7015  15.1101  14.9128  16.9799
2015-12-29   17.8270  23.3871  25.3226  26.8905  15.2319  14.9137  16.9150
2015-12-30   17.8066  23.3216  25.2267  26.7707  15.1556  14.9146  16.9249
2015-12-31   17.7733  23.2085  25.0635  26.5715  15.0263  14.9154  16.9549

             C Fund   S Fund   I Fund
date
2015-10-01  25.7953  34.0993  23.3202  NaN
2015-10-02  26.1669  34.6504  23.6367
2015-10-05  26.6467  35.3565  24.1475
2015-10-06  26.5513  35.1320  24.2294
2015-10-07  26.7751  35.6035  24.3671
2015-10-08  27.0115  35.9016  24.6406
2015-10-09  27.0320  35.9772  24.7723
...             ...      ...      ...  ...
2015-12-22  27.4848  35.0903  23.8679
2015-12-23  27.8272  35.5749  24.3623
2015-12-24  27.7831  35.6084  24.3272
2015-12-28  27.7230  35.4625  24.2816
2015-12-29  28.0236  35.8047  24.4757
2015-12-30  27.8239  35.5126  24.4184
2015-12-31  27.5622  35.2356  24.0952

[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'])
    Nasdaq Traded                                                        True
    Security Name       International Business Machines Corporation Co...
    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
```

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  111840144.0
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  26458.25  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  \
TRADEDATE
2017-07-03    CNGD  USDRUB_TOM  USD000UTSTOM  58.98  58.840  59.4250  59.3600
2017-07-04    CETS  USDRUB_TOM  USD000UTSTOM  59.30  59.135  59.4575  59.4125
2017-07-04    CNGD  USDRUB_TOM  USD000UTSTOM  59.36  58.930  59.3600  59.3575
2017-07-05    CETS  USDRUB_TOM  USD000UTSTOM  59.30  59.300  60.2600  59.9825
2017-07-05    CNGD  USDRUB_TOM  USD000UTSTOM  59.34  59.265  60.1800  60.1800

            NUMTRADES        VOLRUR  WAPRICE
TRADEDATE
2017-07-03         24  1.864785e+09      NaN
2017-07-04      21053  1.090265e+11  59.2700
2017-07-04         37  1.046416e+09      NaN
2017-07-05      50108  2.874226e+11  59.9234
2017-07-05         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.

```
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                 Traceback (most recent call last)
<ipython-input-8-7059d99fc9b4> in <module>()
----> 1 f = web.DataReader("F", 'yahoo', start, end, session=session)

~/checkouts/readthedocs.org/user_builds/pandas-datareader/envs/stable/lib/python3.5/
↪site-packages/pandas_datareader-0.6.0-py3.5.egg/pandas_datareader/data.py in␣
↪DataReader(name, data_source, start, end, retry_count, pause, session, access_key)
    289     """
    290     if data_source == "yahoo":
--> 291         raise ImmediateDeprecationError(DEP_ERROR_MSG.format('Yahoo Daily'))
    292         return YahooDailyReader(symbols=name, start=start, end=end,
    293                                 adjust_price=False, chunksize=25,

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


In [9]: f.ix['2010-01-04']
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
↪---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-f76d00c0d565> in <module>()
----> 1 f.ix['2010-01-04']

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

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.

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)

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 indicies of each series.

Return type DataFrame

url
API URL

4.5.2 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

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.

4.5.3 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

params
Parameters to use in API calls

read()
Read data from connector

url
API URL

4.5.4 Engima

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

```
>>> 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

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 intiating 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

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.

close()
Close network session

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=0.001*, *session=None*)
Symbols available for trading on IEX

Notes

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

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 intiating 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.

`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 divdends 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

close()
Close network session

params
Parameters to use in API calls

read()
Read data

url
API URL

4.5.9 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

4.5.10 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

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

close()
Close network session

params
Parameters to use in API calls

read()
Read data

url
API URL

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 intiating pause.
- **`session`** (*`Session, default None`*) – requests.sessions.Session instance to be used

Notes

See Stooq

close()
Close network session

params
Parameters to use in API calls

read()
Read data

url
API URL

4.5.14 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 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*.

close()
Close network session

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 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.

close()
Close network session

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 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*.

close()
Close network session

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

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

close()
Close network session

params
Parameters to use in API calls

read()
read one data from specified URL

url
API URL

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.*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, 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.*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:

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index