New issue

Jump to bottom

Getting benckmark data via IEX API does not work anymore #2480

Closed MarikoKujo opened this issue on Jun 16, 2019 · 55 comments

Labels Benchmark

MarikoKujo commented on Jun 16, 2019

Zipline uses IEX API to get benchmark data in `benchmarks.py` :

```
def get_benchmark_returns(symbol):
    """
    Get a Series of benchmark returns from IEX associated with `symbol`.
    Default is `SPY`.

    Parameters
    ----------
    symbol : str
        Benchmark symbol for which we're getting the returns.

    The data is provided by IEX (https://iextrading.com/), and we can
    get up to 5 years worth of data.
    """
    r = requests.get(
        'https://api.iextrading.com/1.0/stock/{}/chart/5y'.format(symbol)
    )
    data = r.json()

    df = pd.DataFrame(data)

    df.index = pd.DatetimeIndex(df['date'])
    df = df['close']
```

However, according to the IEX FAQ page, the chart api was already removed on June 15, 2019. Currently, using this api to try to download any stock data such as SPY will return nothing but an HTTP 403 error. The functions of deprecated APIs are now transferred to their new API, IEX Cloud, which requires a unique token per user in any request.
Any idea how to fix this issue in the long run?

SeungYong-Baek mentioned this issue on Jun 17, 2019

zipline 1.3.0 JSONDecodeError #2481 Open

ehtycuact commented on Jun 17, 2019

Same issue here. Looking for a way to stop this whole mechanics either by providing the data from another source (a file, etc.) or just remove it completely.

tibkiss commented on Jun 17, 2019 Contributor

IEX has deprecated the old API in favor of the new one:
https://iextrading.com/developer/#sunset-schedule

iexcloud.io offers the chart endpoint used in zipline, but user registration
is required to obtain a token for the service.

tibkiss commented on Jun 17, 2019 Contributor

My temporal fix (on zipline-live, which is branched off from 1.1.0):

```
diff --git a/zipline/data/benchmarks.py b/zipline/data/benchmarks.py
index 45137428..72d9c3cc 100644
--- a/zipline/data/benchmarks.py
+++ b/zipline/data/benchmarks.py
@@ -30,8 +30,10 @@ def get_benchmark_returns(symbol):
     The data is provided by IEX (https://iextrading.com/), and we can
     get up to 5 years worth of data.
     """
+    IEX_TOKEN = 'pk_TOKEN_COMES_HERE'  # FIXME: move to param
     r = requests.get(
-        'https://api.iextrading.com/1.0/stock/{}/chart/5y'.format(symbol)
+        'https://cloud.iexapis.com/stable/stock/{}/chart/5y?token={}'.format(
```

👍 14 👎 2 😄 3

MarikoKujo commented on Jun 17, 2019 Author

Inspired by this comment in #1951, here is the workaround for people who does NOT need benchmarks at all:

By default, zipline downloads benchmark data by making an http request in `get_benchmark_returns()` in `zipline/data/benchmarks.py`. It returns a `pd.Series` which will be saved to a csv file by `ensure_benchmark_data()` in `zipline/data/loaders.py`. So we can create a dummy benchmark file by setting all data entries to zero.

First, replace `benchmarks.py` with:

```
import pandas as pd
from trading_calendars import get_calendar

def get_benchmark_returns(symbol, first_date, last_date):
    cal = get_calendar('NYSE')

    dates = cal.sessions_in_range(first_date, last_date)

    data = pd.DataFrame(0.0, index=dates, columns=['close'])
    data = data['close']

    return data.sort_index().iloc[1:]
```

Then in `loaders.py`, replace
`data = get_benchmark_returns(symbol)`
with
`data = get_benchmark_returns(symbol, first_date, last_date)`

In this example `NYSE` is used, but it also works when I use `AlwaysOpenCalendar` in my backtest, so I did not try to change it to some other calendar.

This is only a hack. In the long run I would suggest to change the benchmark downloading method to request other API in case you would like to use benchmarks in the future.

👍 21 🎉 5 ❤️ 8

SeungYong-Baek commented on Jun 18, 2019

Today morning, I did the test with your hack.
Unfortunately, It doesn't worked in my environment. I haven't found the cause yet.
I am going to try again with your hack carefully.
Thank you very much.

@tibkiss
I did the test with your fix. After fail, I noticed your environment is very different from my environment.
Thank you very much.

shlomikushchi commented on Jun 18, 2019 • edited

here's a fix that goes back to yahoo as a benchmark source.
replace this method in benchmarks.py and don't forget to change the call to it in loader.py

```
import numpy as np
import pandas as pd
import pandas_datareader.data as pd_reader

def get_benchmark_returns(symbol, first_date, last_date):
    """
    Get a Series of benchmark returns from Yahoo associated with `symbol`.
    Default is `SPY`.

    Parameters
    ----------
    symbol : str
        Benchmark symbol for which we're getting the returns.

    The data is provided by Yahoo Finance
    """
    data = pd_reader.DataReader(
        symbol,
        'yahoo',
        first_date,
        last_date
    )

    data = data['Close']

    data[pd.Timestamp('2008-12-15')] = np.nan
    data[pd.Timestamp('2009-08-11')] = np.nan
    data[pd.Timestamp('2012-02-02')] = np.nan

    data = data.fillna(method='ffill')

    return data.sort_index().tz_localize('UTC').pct_change(1).iloc[1:]
```

SeungYong-Baek commented on Jun 18, 2019

@MarikoKujo
@shlomikushchi

I did the test with MarikoKujo's and shlomikushchi 's temporary fix.
Unfortunately, It doesn't worked in my environment. Two fixes showed the same error.
I think that the cause of error is my different environment from yours.
So, I need to investigate more my environment.
Thank you so much.

/home/seungyong/zipline/lib/python3.5/site-packages/empyrical/stats.py:704: RuntimeWarning: invalid value encountered in true_divide
out=out,
/home/seungyong/zipline/lib/python3.5/site-packages/empyrical/stats.py:790: RuntimeWarning: invalid value encountered in true_divide
np.divide(average_annual_return, annualized_downside_risk, out=out)
Traceback (most recent call last):
File "pandas/_libs/index.pyx", line 449, in pandas._libs.index.DatetimeEngine.get_loc
File "pandas/_libs/hashtable_class_helper.pxi", line 811, in pandas._libs.hashtable.Int64HashTable.get_item
File "pandas/_libs/hashtable_class_helper.pxi", line 817, in pandas._libs.hashtable.Int64HashTable.get_item
KeyError: 1421625600000000000

During handling of the above exception, another exception occurred:

```
File "/home/seungyong/zipline/lib/python3.5/site-packages/zipline/algorithm.py", line 756, in run
for perf in self.get_generator():
File "/home/seungyong/zipline/lib/python3.5/site-packages/zipline/gens/tradesimulation.py", line
209, in transform
for capital_change_packet in once_a_day(dt):
File "/home/seungyong/zipline/lib/python3.5/site-packages/zipline/gens/tradesimulation.py", line
160, in once_a_day
algo.data_portal,
File "/home/seungyong/zipline/lib/python3.5/site-packages/zipline/finance/metrics/tracker.py", line
272, in handle_market_open
session_label,
File "/home/seungyong/zipline/lib/python3.5/site-packages/zipline/finance/metrics/tracker.py", line
61, in execution_open_and_close
open, close = calendar.open_and_close_for_session(session)
File "/home/seungyong/zipline/lib/python3.5/site-packages/trading_calendars/trading_calendar.py",
line 760, in open_and_close_for_session
sched.at[session_label, 'market_open'].tz_localize(UTC),
File "/home/seungyong/zipline/lib/python3.5/site-packages/pandas/core/indexing.py", line 1869, in
getitem
return self.obj._get_value(*key, takeable=self._takeable)
File "/home/seungyong/zipline/lib/python3.5/site-packages/pandas/core/frame.py", line 1985, in
_get_value
return engine.get_value(series._values, index)
File "pandas/_libs/index.pyx", line 83, in pandas._libs.index.IndexEngine.get_value
File "pandas/_libs/index.pyx", line 91, in pandas._libs.index.IndexEngine.get_value
File "pandas/_libs/index.pyx", line 451, in pandas._libs.index.DatetimeEngine.get_loc
KeyError: Timestamp('2015-01-19 00:00:00+0000', tz='UTC')
```

zipper-123 commented on Jun 20, 2019

This seems to have a few issues going now so Ill weigh in on the most commented one.

I tried all of the workarounds with no luck.

Replacing the get_bencmark_returns() method just defers to other null references down the line.

Fixing the IEX API defers to the treasury data failing for me. Since I am using BTC data, OHLCV.

Is there any supported way to backtest crypto currency data? Are the benchmark and treasury data pulls really that crucial?

@zipper-123 Perhaps you have a faulty cached file. Remember that ensure_benchmark_data in loader.py first attempts to read from disk. That happened to my while I was tinkering with a solution.

I ended up implementing a minor variation of the solution suggested by **@marketneutral** to sidestep the issue until it's properly fixed. I kept the signature of get_benchmark_returns, and just used a wider date range than I'll ever need.

In benchmark_py

```
def get_benchmark_returns(symbol):
    cal = get_calendar('NYSE')
    first_date = datetime(1930,1,1)
    last_date = datetime(2030,1,1)
    dates = cal.sessions_in_range(first_date, last_date)
    data = pd.DataFrame(0.0, index=dates, columns=['close'])
    data = data['close']
    return data.sort_index().iloc[1:]
```

And bypassing the cache in loader.py

```
    """
    if data is not None:
        return data
    """
```

👍 5

grobbie94 commented on Jun 27, 2019 • edited

Another fix is getting a free IEX api key and altering the API request in benchmarks.py

```
r= requests.get(
        "https://cloud.iexapis.com/stable/stock/{}/chart/5y?chartCloseOnly=True&token=
{}".format(symbol, IEX_KEY)
    )
```

👍 15 😄 4

MarikoKujo mentioned this issue on Jun 28, 2019

backtest fails trying to download SPY data for benchmark with quandl Open

SeungYong-Baek commented on Jul 2, 2019

@tibkiss
@grobbie94
Thank you for your fix. It works well for my script.

ElliotVilhelm commented on Jul 9, 2019

jesus christ .. just spent two hours on this

cddventura commented on Jul 18, 2019

Has the solution been found for this?

shlomikushchi commented on Jul 18, 2019

yes... mine is not a temporary fix.
it's a solution. I'm working with it for more than a month

👍 2 😕 2

mikebardpython commented on Jul 24, 2019

getting the same error, tried all the solutions on this thread as well as the one here

is there a solution forthcoming?

weu39843 commented on Jul 29, 2019

Getting the same error using local data (csv -> DataFrame -> panel) and it's not clear to me how your fix works **@shlomikushchi** as I'm not using yahoo data. Maybe you have an idea how to tweak your code in order to fix the problem when using local data.

This is still not working. **@weu39843** I'm having the same issue with local data as well, passing a CSV as a DataFrame. When trying **@AndreasClenow** solution and all the others I get the following error:

```
Traceback (most recent call last):
 File "<stdin>", line 39, in <module>
 File "/usr/local/lib/python3.6/dist-packages/zipline/utils/run_algo.py", line 430, in
run_algorithm
   blotter=blotter,
 File "/usr/local/lib/python3.6/dist-packages/zipline/utils/run_algo.py", line 188, in _run
   trading_days=trading_calendar.schedule[start:end].index,
 File "/usr/local/lib/python3.6/dist-packages/zipline/finance/trading.py", line 103, in
__init__
   self.bm_symbol,
 File "/usr/local/lib/python3.6/dist-packages/zipline/data/loader.py", line 149, in
load_market_data
   environ,
 File "/usr/local/lib/python3.6/dist-packages/zipline/data/loader.py", line 201, in
ensure_benchmark_data
   environ)
 File "/usr/local/lib/python3.6/dist-packages/zipline/data/loader.py", line 324, in
_load_cached_data
   data = from_csv(path)
 File "/usr/local/lib/python3.6/dist-packages/zipline/data/loader.py", line 307, in
from_csv
   squeeze=True,
 File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 562, in parser_f
   return _read(filepath_or_buffer, kwds)
 File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 325, in _read
   return parser.read()
 File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 815, in read
   ret = self._engine.read(nrows)
 File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 1387, in read
   index, names = self._make_index(data, alldata, names)
 File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 1030, in
_make_index
   index = self._agg_index(index)
 File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 1111, in
_agg_index
   arr = self._date_conv(arr)
 File "/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py", line 2272, in
converter
   lib.try_parse_dates(strs, dayfirst=dayfirst))
 File "/usr/local/lib/python3.6/dist-packages/pandas/util/decorators.py", line 91, in
wrapper
   return func(*args, **kwargs)
 File "/usr/local/lib/python3.6/dist-packages/pandas/tseries/tools.py", line 291, in
to_datetime
   unit=unit, infer_datetime_format=infer_datetime_format)
 File "/usr/local/lib/python3.6/dist-packages/pandas/tseries/tools.py", line 427, in
_to_datetime
   return _convert_listlike(arg, box, format)
 File "/usr/local/lib/python3.6/dist-packages/pandas/tseries/tools.py", line 398, in
```

```
  File "pandas/tslib.pyx", line 2269, in pandas.tslib.array_to_datetime
(pandas/tslib.c:40684)
  File "pandas/tslib.pyx", line 2229, in pandas.tslib.array_to_datetime
(pandas/tslib.c:39919)
  File "pandas/src/datetime.pxd", line 141, in datetime._string_to_dts
(pandas/tslib.c:85631)
SystemError: <class 'str'> returned a result with an error set

<class 'str'> returned a result with an error set
```

AndreasClenow commented on Aug 5, 2019

@jecker7 when you say passing a local csv, do you mean that you provide a **DataPanel** as the *data* parameter for *zipline.run_algorith*?

That way of providing data is no longer supported and Q are strongly recommending against using it. I didn't try that method in quite some time, after they told me a couple of years ago not to use it.

I'd suggest using the regular bundle/ingest process to read your csv.

sinnergarden commented on Aug 8, 2019

@shlomikushchi I tried your solution, it works well for me!! Thanks

BTW, what is the point of doing the following operations:

```
    data[pd.Timestamp('2008-12-15')] = np.nan
    data[pd.Timestamp('2009-08-11')] = np.nan
    data[pd.Timestamp('2012-02-02')] = np.nan
```

shlomikushchi commented on Aug 8, 2019

@sinnergarden glad to hear that
it was copied from the original zipline code.
before zipline moved to IEX they used pd.DataReader. they used google and not yahoo.
but I copied the code from their solution and switched to yahoo.

Zipline will have to move their datasource for SPY data somewhere else besides IEX, or change the setup to require an API key. Another way to fix is:

- sign up here: https://iexcloud.io/cloud-login#/register
- find where your benchmarks.py file is: `ipython`, `import zipline`, `zipline.__file__`
- Open the zipline lib in an IDE like Atom
- Add your token from the IEX dashboard (instead of `pk_numbers...` below), and change the requests line to:

```
token = 'pk_numbersnumbersnumbers'
r = requests.get(
    'https://cloud.iexapis.com/stable/stock/{}/chart/5y?token={}'.format(symbol, token)
)
```

I guess it's the nature of the beast with financial data that it's hard to find for free...but annoying.

👍 3 🎉 1

stocktrader8888 commented on Sep 14, 2019

It would make far more sense for the benchmark routines to use a symbol from a bundle rather than any hardcoded external source.

nateGeorge commented on Sep 16, 2019

@stocktrader8888 no free bundles on Quandl have SPY (default benchmark I think). Could get it with the yfinance package, but that requires the latest pandas and zipline can't use latest pandas.

nateGeorge commented on Sep 16, 2019

Plus IEX only has 5y of history for SPY which is pretty weak. yfinance could get everything

AndreasClenow commented on Sep 17, 2019

My take on solution:

2. A hard coded benchmark from a hard coded source makes no sense. Using the SPY as bench doesn't make sense either, in particular since this api call doesn't seem to take dividends into account. If you really need a bm, have the symbol and bundle configurable.

People who are serious enough about backtesting to bother with setting up a local Zipline are not very likely to rely on Yahoo, Quandle, Google or other free sources, and they are very likely to use proper benchmarks instead of price series of an ETF.

👍 12

exportacc mentioned this issue on Sep 19, 2019

get_benchmark_returns() function returns 403 (Forbidden) #2530 Open

DavisOwen commented on Oct 7, 2019

@AndreasClenow

Just curious, what would be a good source for S&P 500 benchmark data, or other benchmarks that you use?

DavisOwen commented on Oct 7, 2019 • edited

I like Andreas' suggestion, namely that not having benchmark data should be a warning rather than an error. There should also, as he said, be an option for users to provide their own csv/dataframe as benchmark data.

Looking at the code, the TradingEnvironment object is where the benchmark returns are ultimately loaded. I'm not sure where TradingEnvironment gets used by zipline, but I could Imagine a good way to do this would be to add a class method setter to TE where you just provide a benchmark dataframe manually. This seems like it would require the least overhead.

However, referencing the first suggestion, I'm not sure what the expected behavior of zipline should be if there is no benchmark data found. It could be non-trivial to run a test without a benchmark. A workaround to this has been discussed above, where you can simply pass a dataframe of all zeros, but this certainly isn't elegant either. It would be nice to be able to circumvent this entirely, but I think for now the best solution is find some sort of consistent (albeit low quality) data source for zipline to use (like google, yahoo, etc.)

EDIT: It actually looks like a commit was made that addresses this issue (intentionally or not). It's not on the current pip or conda repos for zipline. But it is on the master branch of zipline.

ce2a236

@ssanderson added an argument to run_algorithm called benchmark_returns. If you just run your algorithm with run_algorithm(), you can dowload your own benchmark in csv, run pd.read_csv() to load, and pass it to run_algorithm as benchmark_data. The only issue I'm running into now is I'm not sure the exact format that benchmark_data expects. Looking at _load_cache_data in loader.py, it looksl like this is the command you should run to read the csv, but I'm still having some issues.

```
pd.read_csv(
                path,
                parse_dates=[0],
                index_col=0,
                header=None,
                # Pass squeeze=True so that we get a series instead of a frame.
                squeeze=True,
            ).tz_localize('UTC')
```

If anyone knows the proper formatting for benchmark data we can basically close this issue as people can just provide their own benchmark on a per-test basis.

EDIT 2:

I got it to work. In the process I discovered that there has been an attempt to allow custom benchmark data via creating a bundle with csvdir and adding it to set_benchmarks in initialize.

https://stackoverflow.com/questions/44199678/how-to-manually-provide-a-benchmark-in-zipline

https://www.zipline.io/bundles.html

However, I don't like this because it requires you to have a specific ohlcvs data and seems like a lot of overhead.

If you want a custom benchmark, just use the current master branch of zipline and pass benchmark_returns as an argument to run_algorithm. The format of benchmark returns is a pandas series with the adj. close (or whatever data you want) as the data and the Timestamp as the index.

6 hidden items
Load more...

@zoakes : the solution I posted works for me in production for months now.
Check this out: #2480 (comment)

zoakes commented on Nov 20, 2019

I tried a few of these — including the ones changing Benchmarks.py in libs, no luck so far.

What solution worked in production code ?

Zach

Get Outlook for iOS<https://aka.ms/o0ukef>
…

caot commented on Nov 27, 2019

#2488 is related.

caot commented on Nov 27, 2019 • edited

> My temporal fix (on zipline-live, which is branched off from 1.1.0):
>
> ```
> diff --git a/zipline/data/benchmarks.py b/zipline/data/benchmarks.py
> index 45137428..72d9c3cc 100644
> --- a/zipline/data/benchmarks.py
> +++ b/zipline/data/benchmarks.py
> @@ -30,8 +30,10 @@ def get_benchmark_returns(symbol):
>      The data is provided by IEX (https://iextrading.com/), and we can
>      get up to 5 years worth of data.
>      """
> +    IEX_TOKEN = 'pk_TOKEN_COMES_HERE'  # FIXME: move to param
>      r = requests.get(
> -        'https://api.iextrading.com/1.0/stock/{}/chart/5y'.format(symbol)
> +        'https://cloud.iexapis.com/stable/stock/{}/chart/5y?token={}'.format(
> +            symbol, IEX_TOKEN)
>      )
>      data = json.loads(r.text)
> ```

The 'Publishable' Token can be used from https://iexcloud.io/

> **@zipper-123** Perhaps you have a faulty cached file. Remember that ensure_benchmark_data in loader.py first attempts to read from disk. That happened to my while I was tinkering with a solution.
>
> I ended up implementing a minor variation of the solution suggested by **@marketneutral** to sidestep the issue until it's properly fixed. I kept the signature of get_benchmark_returns, and just used a wider date range than I'll ever need.
>
> In benchmark_py
>
> ```
> def get_benchmark_returns(symbol):
>     cal = get_calendar('NYSE')
>     first_date = datetime(1930,1,1)
>     last_date = datetime(2030,1,1)
>     dates = cal.sessions_in_range(first_date, last_date)
>     data = pd.DataFrame(0.0, index=dates, columns=['close'])
>     data = data['close']
>     return data.sort_index().iloc[1:]
> ```
>
> And bypassing the cache in loader.py
>
> ```
>     """
>     if data is not None:
>         return data
>     """
> ```

how can i bypass the cache? Where exactly should i paste the code? Sorry just learning and thank you.

carstenf commented on Dec 9, 2019

@shlomikushchi

looks like your permanent solution is working for me.
Just a question, you wrote "don't forget to change the call to it in loader.py"
Do i have to change something in the original loader.py file? and if yes -> what do I need to change?

Thanks

bernino commented on Dec 21, 2019

```
diff --git a/zipline/data/benchmarks.py b/zipline/data/benchmarks.py
index 45137428..72d9c3cc 100644
--- a/zipline/data/benchmarks.py
+++ b/zipline/data/benchmarks.py
@@ -30,8 +30,10 @@ def get_benchmark_returns(symbol):
     The data is provided by IEX (https://iextrading.com/), and we can
     get up to 5 years worth of data.
     """
+    IEX_TOKEN = 'pk_TOKEN_COMES_HERE'  # FIXME: move to param
     r = requests.get(
-        'https://api.iextrading.com/1.0/stock/{}/chart/5y'.format(symbol)
+        'https://cloud.iexapis.com/stable/stock/{}/chart/5y?token={}'.format(
+            symbol, IEX_TOKEN)
     )
     data = json.loads(r.text)
```

This resolved it for me (had to +import json though).

Any chance this will be in the official release?

👍 1

richafrank added the **Benchmark** label on Jan 7

samatix mentioned this issue on Jan 9

run_algorithm() fails because get_benchmark_returns() cannot fetch data from api.iextrading.com (403 Forbidden) #2607 Closed

carstenf commented on Jan 11

> here's a fix that goes back to yahoo as a benchmark source.
> replace this method in benchmarks.py and don't forget to change the call to it in loader.py

```
import numpy as np
import pandas as pd
import pandas_datareader.data as pd_reader

def get_benchmark_returns(symbol, first_date, last_date):
    """
    Get a Series of benchmark returns from Yahoo associated with `symbol`.
    Default is `SPY`.
```

```
            Benchmark symbol for which we're getting the returns.

        The data is provided by Yahoo Finance
        """
        data = pd_reader.DataReader(
            symbol,
            'yahoo',
            first_date,
            last_date
        )

        data = data['Close']

        data[pd.Timestamp('2008-12-15')] = np.nan
        data[pd.Timestamp('2009-08-11')] = np.nan
        data[pd.Timestamp('2012-02-02')] = np.nan

        data = data.fillna(method='ffill')

        return data.sort_index().tz_localize('UTC').pct_change(1).iloc[1:]
```

Hi

I changed the benchmark.py and loader.py accordingly....

Inside the code I set:

def initialize(context):
set_benchmark(symbol('SPY'))

which gives an error: 'Symbol 'SPY' was not found.'

using 'AAPL' as anSymbol gives no error.

Than

returns, positions, transactions, benchmark_rets = pf.utils.extract_rets_pos_txn_from_zipline(perf)
pf.create_returns_tear_sheet(returns, benchmark_rets)

gives again an error: ValueError: not enough values to unpack (expected 4, got 3)

so it did not find: benchmark_rets

I think I'm using this wrong with zipline.run_algorithm
How is the correct usage, I'm googling around since hours but can't find an example to use zipline.run_algorithm with a benchmark, basically the original solution.

Could someone please share the correct usage.
Thank you very much

@carstenf you don't need to call `set_benchmark(symbol('SPY'))` it's a default in zipline.
the bundle you use doesn't include SPY so it's not a known symbol

carstenf commented on Jan 13

@shlomikushchi

ok, understood, thanks.

I'm not getting any more errors.
I get a benchmark if I use set_benchmark(symbol('SPY')), because I included it now in the bundle.

But I don't get a benchmark if I want to pull it from benchmark.py.

I did the changes in benchmark.py(copied you fix), and I changed one line in the loader.py:
#data = get_benchmark_returns(symbol)
data = get_benchmark_returns(symbol,first_date, last_date).
....should be ok...but no benchmark...something missing?

and were is the benchmark symbol defined?
Might be a good idea to use ^SP500TR, it's the total return of the S&P500, as Andreas suggested.

thanks

shlomikushchi commented on Jan 14 • edited

benchmark symbol is defined in loader.py: `load_market_data()`

ssanderson commented on Jan 21 Contributor

Hey all. I'm taking a look at this this morning.

ssanderson mentioned this issue on Jan 21

Remove Implicit Dependency on Benchmarks and Treasury Returns #2627 Open

We've got a bunch of issues related to the IEX-sourced benchmark data failing. In the interest of having a single canonical location for the issue, I'm closing this in favor of the newly added #2627. Feel free to comment there if there's information I've missed from this thread.

ssanderson closed this on Jan 21

sislate mentioned this issue on Feb 26

KeyError: Equity(0 [SBER]) with custom bundle from csv #2661 Open

awesomefrank commented on Mar 4

> here's a fix that goes back to yahoo as a benchmark source.
> replace this method in benchmarks.py and don't forget to change the call to it in loader.py
>
> ```
> import numpy as np
> import pandas as pd
> import pandas_datareader.data as pd_reader
>
> def get_benchmark_returns(symbol, first_date, last_date):
>     """
>     Get a Series of benchmark returns from Yahoo associated with `symbol`.
>     Default is `SPY`.
>
>     Parameters
>     ----------
>     symbol : str
>         Benchmark symbol for which we're getting the returns.
>
>     The data is provided by Yahoo Finance
>     """
>     data = pd_reader.DataReader(
>         symbol,
>         'yahoo',
>         first_date,
>         last_date
>     )
>
>     data = data['Close']
>
>     data[pd.Timestamp('2008-12-15')] = np.nan
>     data[pd.Timestamp('2009-08-11')] = np.nan
>     data[pd.Timestamp('2012-02-02')] = np.nan
> ```

Thank you **@shlomikushchi**
The solution works very well for me.

zoakes commented on Mar 4

I'm not able to find loader.py file — hows that possible ?

Anyone have a path to it by chance ?

Zach

Get Outlook for iOS<https://aka.ms/o0ukef>
…

awesomefrank commented on Mar 4

> ________________________________ From: awesomefrank notifications@github.com Sent: Wednesday, March 4, 2020 12:13:56 PM To: quantopian/zipline zipline@noreply.github.com Cc: Zach Mazz zach_oakes@outlook.com; Mention mention@noreply.github.com Subject: Re: [quantopian/zipline] Getting benckmark data via IEX API does not work anymore (#2480) here's a fix that goes back to yahoo as a benchmark source. replace this method in benchmarks.py and don't forget to change the call to it in loader.py import numpy as np import pandas as pd import pandas_datareader.data as pd_reader def get_benchmark_returns(symbol, first_date, last_date): """ Get a Series of benchmark returns from Yahoo associated with `symbol` . Default is `SPY` . Parameters ---------- symbol : str Benchmark symbol for which we're getting the returns. The data is provided by Yahoo Finance """ data = pd_reader.DataReader(symbol, 'yahoo', first_date, last_date) data = data['Close'] data[pd.Timestamp('2008-12-15')] = np.nan data[pd.Timestamp('2009-08-11')] = np.nan data[pd.Timestamp('2012-02-02')] = np.nan data = data.fillna(method='ffill') return data.sort_index().tz_localize('UTC').pct_change(1).iloc[1:] Thank you **@shlomikushchi**https://eur04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fshlomikushchi&data=02%7C01%7C%7C681668adb70445608d1108d7c067ce5c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C637189424377455029&sdata=xUqug511h3s2xt3m%2FTsoKCPRs%2F1afHK2zq4vC4g8Y78%3D&reserved=0 The solution works very well for me. — You are receiving this because you were mentioned. Reply to this email directly, view it on GitHubhttps://eur04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fquantopian%2Fzipline%2Fissues%2F2480%3Femail_source%3Dnotifications%26email_token%3DALPUCIETFFDQUCB2KWAOZPTRF2K6JA5CNFSM4HYSHEN2YY3PNVWWK3TUL52HS4DFVREXG43VMVBW63LNMVXHJKTDN5WW2ZLOORPWSZGOENZLJ6Y%23issuecomment-594719995&data=02%7C01%7C%7C681668adb70445608d1108d7c067ce5c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C637189424377455029&sdata=Zm1GtpfLOEmgNCuHF9%2BqorCtkJz648H51VAY4BNqJoo%3D&reserved=0, or unsubscribehttps://eur04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fnotifications%2Funsubscribe-auth%2FALPUCID4BLTZ7OLNF7QNJPLRF2K6JANCNFSM4HYSHENQ&data=02%7C01%7C%7C681668adb70445608d1108d7c067ce5c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C637189424377465037&sdata=l1DnsxWz5qkcYyy%2FhEtJhkqeCHQzJOoQNRZu1d%2Fjhbk%3D&reserved=0.

..\site-packages\zipline\data

adamhadani commented on Mar 16 • edited

slight variation on one of the posted answers to use the new IEX Cloud API endpoint. this one will read your token from env var `IEX_CLOUD_API_TOKEN` :

```
 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 # See the License for the specific language governing permissions and
 # limitations under the License.
+from os import getenv
+
 import pandas as pd
 import requests


+IEX_CLOUD_API_TOKEN = getenv("IEX_CLOUD_API_TOKEN", "<IEX_CLOUD_API_TOKEN>")
+
+
 def get_benchmark_returns(symbol):
     """
     Get a Series of benchmark returns from IEX associated with `symbol`.
@@ -30,7 +35,7 @@
     get up to 5 years worth of data.
     """
     r = requests.get(
-        'https://api.iextrading.com/1.0/stock/{}/chart/5y'.format(symbol)
+        "https://cloud.iexapis.com/stable/stock/{}/chart/5y?chartCloseOnly=True&token={}".for
     )
     data = r.json()
```

Can apply by saving the above to a file `benchmarks.patch` and then invoke in the directory of the original `benchmarks.py` (e.g. `<virtualenv_root>/lib/python3.5/site-packages/zipline/data/`):

```
$ patch < benchmarks.patch
```

billmoling mentioned this issue on Mar 23

Zipline Jsondecoder error #2669 Open

Sigmares commented on Mar 29 • edited

> The triple quote is a block comment, meaning that everything between will be disabled.
> """
> anything between the triple quotes will be interpreted as a comment, not executed as code.
> print("This will not be printed")
> """
> print("This will be printed")

> It ran the codes somewhat smooth and produced the graphs. However, now I have this notification below the kernel. (In a red box)
>
> "C:\Users\name\Anaconda3\envs\zip35\lib\site-packages\empyrical\stats.py:704:
> RuntimeWarning: invalid value encountered in true_divide
> out=out,
> C:\Users\name\Anaconda3\envs\zip35\lib\site-packages\empyrical\stats.py:790:
> RuntimeWarning: invalid value encountered in true_divide
> np.divide(average_annual_return, annualized_downside_risk, out=out)"
>
> Is this a big problem? and will it cause me further problems down the line?
>
> If anyone could help me understand why it's happening and if there's a solution for it, I'd be greatful.
>
> Thank you.

Hi **@k-s30011**, did you find a solution? I have exactly the same issue.

marlowequart commented on Mar 31 • edited

I am also not able to get this working following the suggestions above. It seems like it is timing out trying to divide, possibly because the benchmark is just a bunch of zeros? I am just trying to run a basic file.

I am getting this same error: RuntimeWarning: invalid value encountered in true_divide np.divide(average_annual_return, annualized_downside_risk, out=out). Does anyone know where this divide is being called from? I don't see it in benchmarks.py or loader.py

I am wondering if it might be due to the version of numpy and other packages I am using. Using numpy version 1.14.6, python 3.5.6

marlowequart commented on Apr 4 • edited

@Sigmares

> Hi **@k-s30011**, did you find a solution? I have exactly the same issue.

above for the API issue, the benchmark data that is loaded is zeros and this is causing the divide function issues. I don't think this is a very good solution, but I cannot find where in the zippline algorithm it is calling the stats functions so this will have to do for now.

Does anyone know where in the zipline package the functions for sharpe and sorting are called? it seems like it would be better to comment those out in zipline rather than the stats module.

CodeMonkeysGottaCode mentioned this issue on Apr 8

Installed Zipline, having issues running and importing it in my Jupyter Notebook. #2675 Open

veekon mentioned this issue 15 days ago

zipline example algo's fail on "JSONDecodeError" using docker image #3 Open

PatrickTunni commented 11 days ago

@k-s30011 **@marlowequart** Looks like I'm late to the game, and a newbie but I've gotten as far as repairing the benchmarks and loader files, got my first backtest to work, albeit with the divide by zero error.

Have you been successful with this onward in Andreas book? I am getting the idea that passing zero's into the code is causing it to throw those errors.

Thanks for the feedback.

cemal95 commented 10 days ago

@PatrickTunni You are better off using your own data, for example if you have your own SPY data you can use it for the benchmark returns. Andreas's is book is pretty nice in terms of programming for beginners and is probably top 5 for algo trading for beginners. Also quantopian lectures slides are really usefull and a good place to start.

If you dont have SPY data, I have data from 1990 to 2020 of data, check my github

Assignees

No one assigned

Benchmark

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

31 participants

and others