Item 1 --- Data ingest: where it comes from and where it goes to

标准库里的ingest函数的使用方式并不是我们期待的，我们更希望将这个功能函数化，流程化，更适用于本地数据转换为zipline回测时可直接应用的数。因此，我们需要先了解一下基本模块。

数据源有哪些？

首先，数据源根据不同使用者不能一概而论，但是对于入门级选手有几个开放数据源可用。一下仅对这几个数据源进行简单介绍。

根据DataReader提供的接口，DataReader所支持的数据源包括：

"yahoo"：yahoo的行情数据

"yahoo-actions"：yahoo的公司行为数据，似乎现在已经不支持了

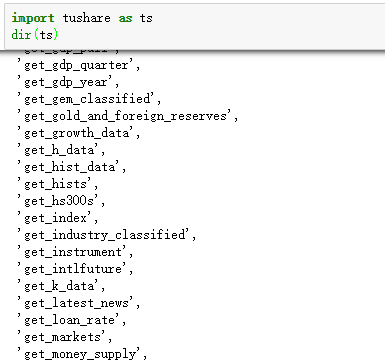
"yahoo-dividends"：yahoo的分红数据

"google"：google的行情数据

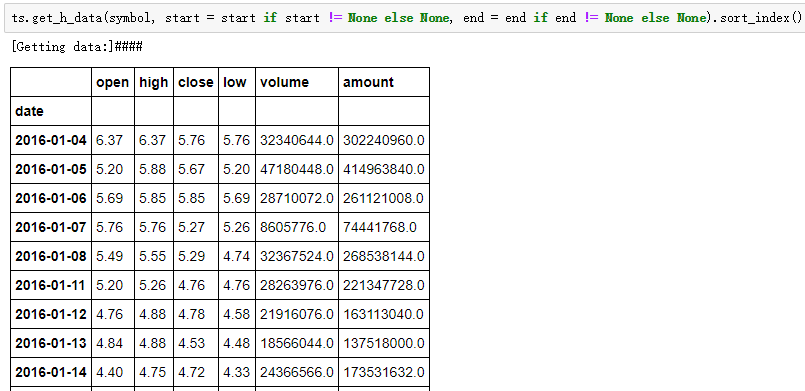
"fred"：FRED的经济研究数据库

另外，tushare有较好的国内A公开数据，接口完善，背后为通联的数据源。

Tushare提供较为丰富的数据接口供用户使用

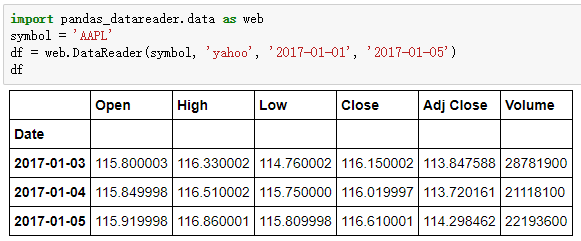


举例说明提取历史行情数据

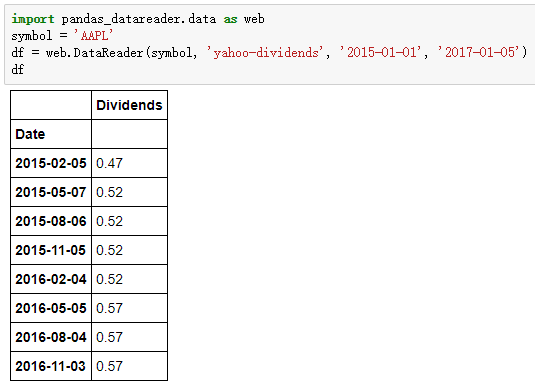


DataReader取自pandas\_datareader模块，该模块为DataFrame结构的远程接入模块，可以通过网络获取相应数据存为DataFrame格式。

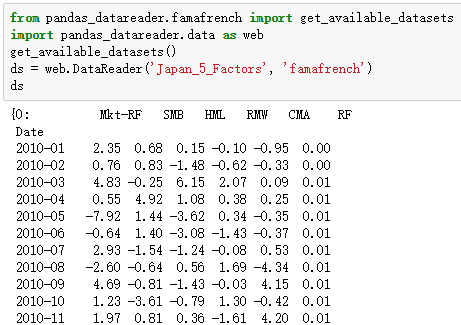
如yahoo股票数据，注：yahoo数据中的close是公司行为调整过的数据，adj close是在close基础上进行dividend调整。



Yahoo分红数据



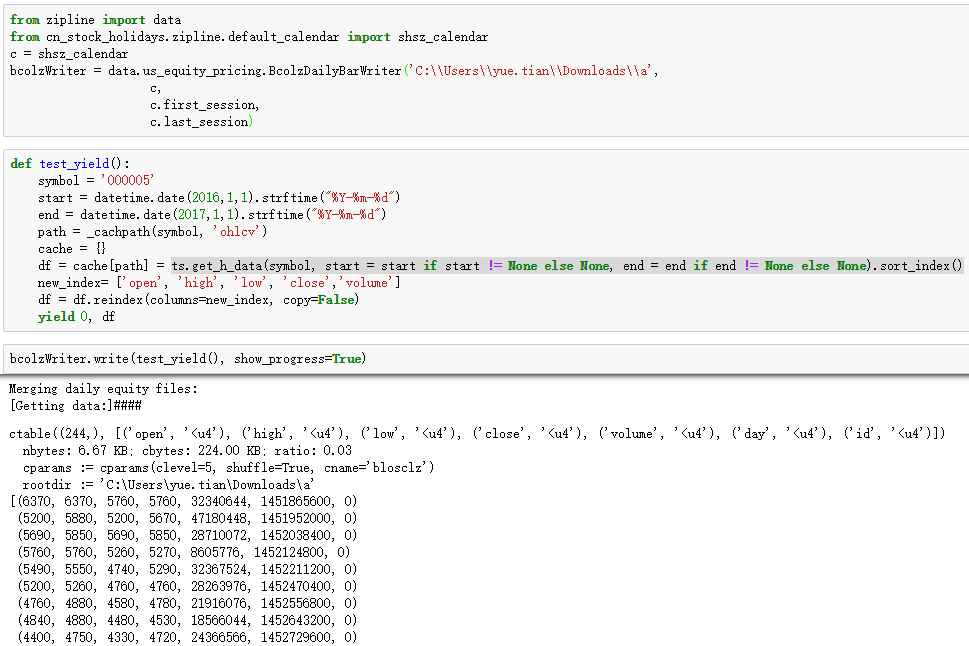
Fama French数据



好了，既然有了数据源，我们的问题是如何将数据源提供给zipline用？按照zipline的格式，我们需要将数据存为bcolz格式供回测使用。

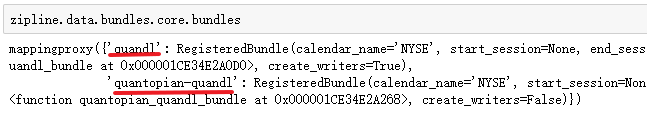
Bcolz数据结构是为了更高效的存储和读取大型数据集，特别是在内存容量有限的情况下，可以存储于硬盘，高效主要源自底层高效的压缩和解压缩操作、底层多线程处理等。

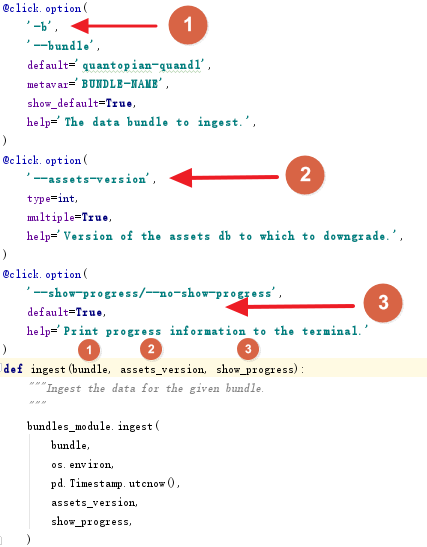
Zipline提供的Bcolz数据接口位于zipline.data.us\_equity\_pricing中，BcolzDailyBarWriter类处理写入操作，其构造函数需要数据存储路径、日历对象（Calendar）、开始和结束日期（session）。这里使用的calendar为沪深日历，其来自cn\_stock\_holidays模块，需要单独安装，开始和结束session应包含数据开始和结束部分。BcolzDailyBarWriter的write函数处理写操作，data参数即为我们数据源读取的数据，但是要注意，数据格式应为可循环结构，要将数据包装成类似[(1,dataFrame1),( 2,dataFrame2),( 2,dataFrame2)]结构。



怎样注入数据：

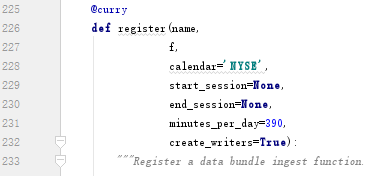
标准做法如下，在\_\_main\_\_.py中定义ingest接口，通过click模块构建命令行操作注入指定bundle，click可以自动生成指定参数类型，默认值，帮助项等，下面的函数定义会在命令行调用，举例命令行执行zipline ingest -b quantopian-quandl，在函数内部bundles\_module会被调用，该模块为我们比较关心修改的部分。bundles\_module是在zipline.data.bundles中定义的bundles实例，其中涉及到register，unregister，clean，ingest等函数。为保持原代码结构完整，我们可以保留用法。zipline.data.bundles.core.bundles中存储了当前已经register的bundle名字和内容，如：





Zipline数据读取存储机制

Zipline的core负责封装真正的执行类，每个数据bundle\_name在注入之前需要register，如下：

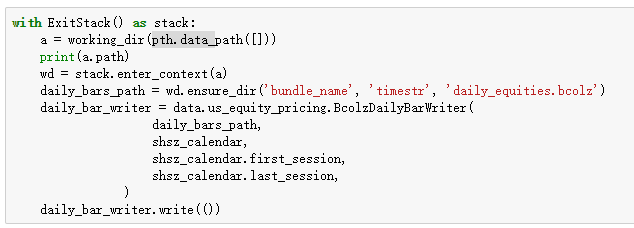


使用如下：



这里的’bundle\_test’为之后在内部dictionary中记录的名称，your\_bundle\_class需要自己定义的类，calendar设置为string名称即可，名称应在TradingCalendarDispatcher中初始化。

Ingest时需要创建BcolzDailyBarWriter对象作为数据写的接口，写的过程中数据会被首先写入临时文件夹，之后转移到指定文件夹。这个过程通过context实现，例如：



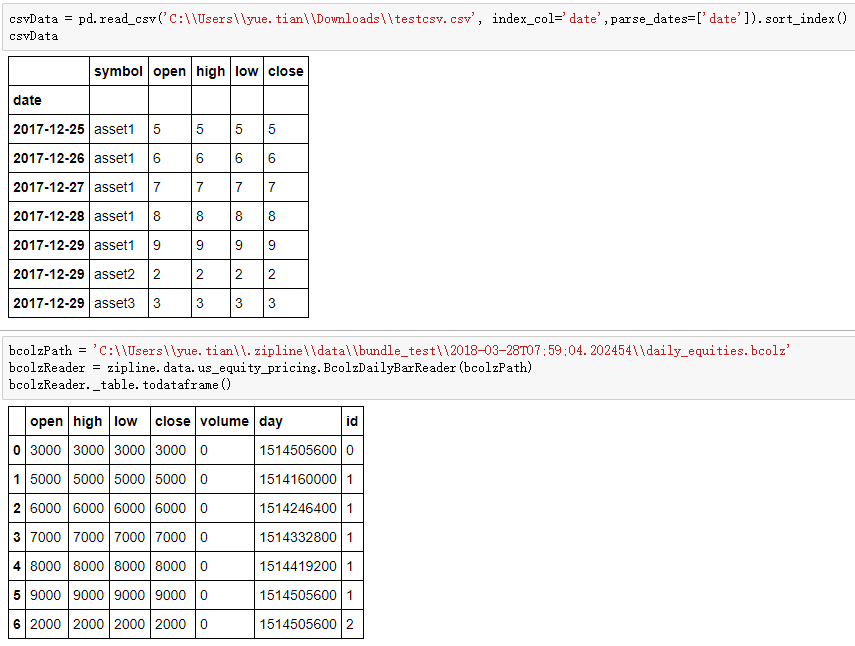
pth.data\_path([])为最终目标文件，在zipline.utils.paths中定义，os.environ中的ZIPLINE\_ROOT可以指定文件夹，如果不指定，当前用户下的.zipline会是默认文件夹expanduser('~/.zipline')。

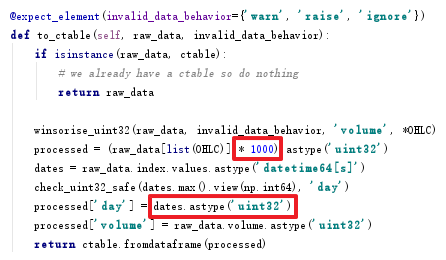
我们主要需要定义ZIPLINE\_ROOT和your\_bundle\_class这个类。

我的例子如下：通过register注册’bundle\_test’，同时关联csv\_bundle，之后通过zipline.data.bundles.ingest接口注入数据。



注意：bcolzWriter在写入数据时会调整原始数据，原始数据乘以1000，日期转为uint32





回测框架

更改文件data/benckmarks.py

更改文件data/loader.py

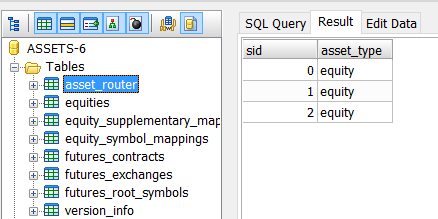
更改文件finance/trading.py

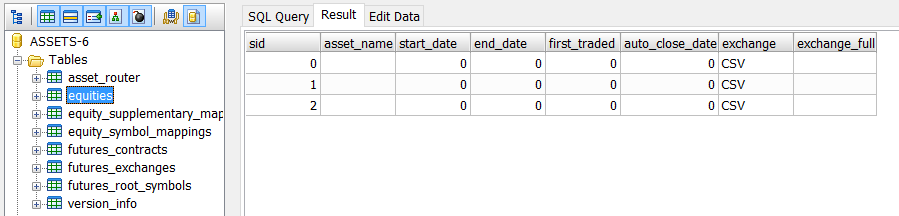
添加文件data/cn\_treasury\_curve.py

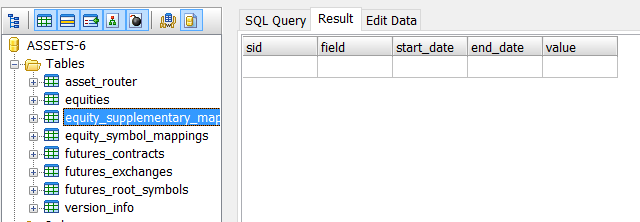
Tradesimulation.py line 220 the for loop

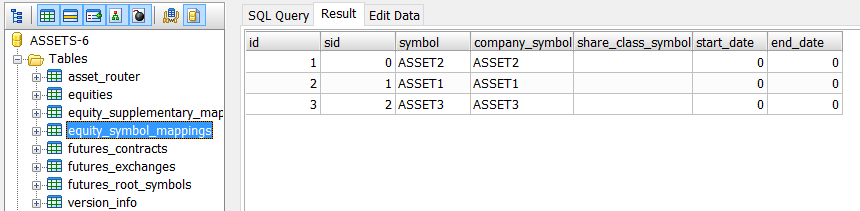
哪些数据库：

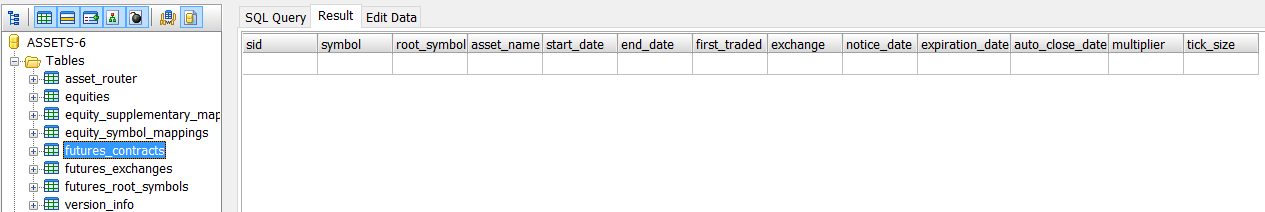
AssetFinder: sqlite数据库assets-6.sqlite中8张表

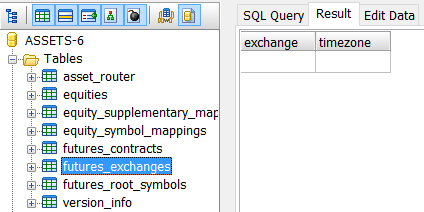


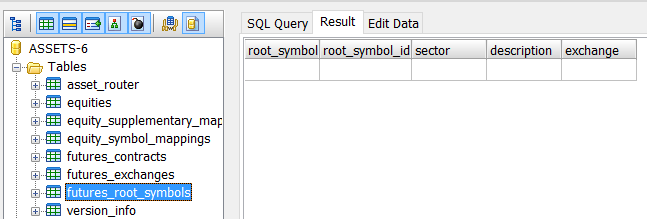


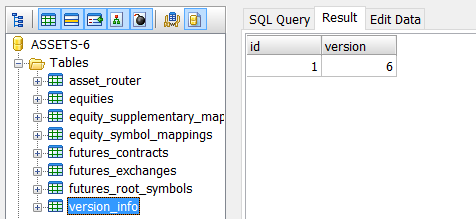








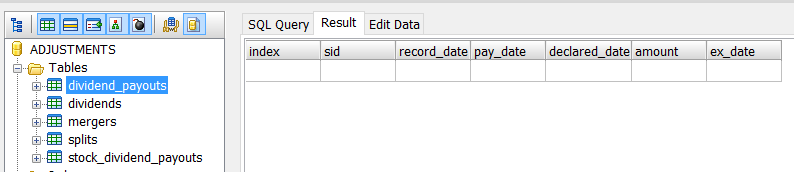


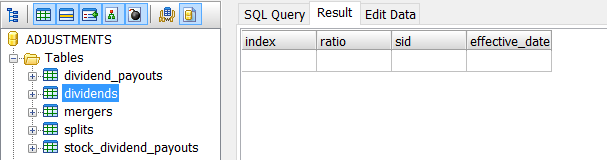


BcolzMinuteBarReader: bcolz数据库minute\_equities.bcolz

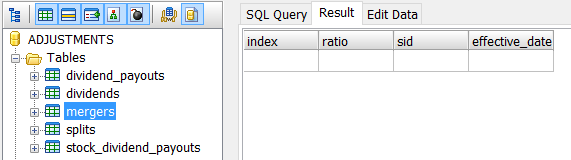
BcolzDailyBarReader: bcolz数据库daily\_equities.bcolz

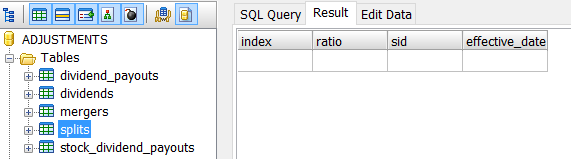
SQLiteAdjustmentReader: sqlite数据库adjustments.sqlite中的5张表格

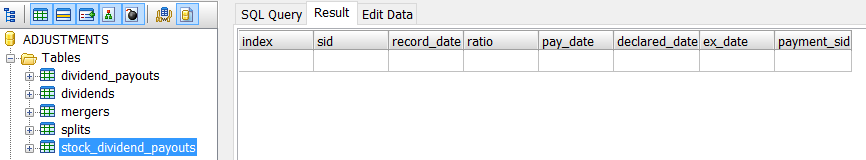




Us\_equity\_pricing.py line 1043 ratio = 1.0 - amount / prev\_close







回测框架详解：

1. Sim\_params

通过create\_simulation\_parameters构造SimulationParameters类对象，对象变量主要包括回测开始时间，结束时间，初始现金，回测频率，交易日历。因此，回测的基本参数均可以在此处设置。

1. Env

TradingEnvironment主要提供benchmark和risk free rate以便计算风险参数。load为可选参数，为函数，需返回双变量bm和tr，格式如下

bm = pd.Series(data=0, index=trading\_days)

tr = pd.DataFrame(data=0, index=trading\_days, columns=['1month', '3month', '6month', '1year', '2year', '3year', '10year'])

如load为空，load\_dump\_data会被调用，返回为0值。可参照data.loader.load\_market\_data函数自行加载本地数据。

1. Initialize

Initialize为重要函数之一，赋值到TradingAlgorithm的\_initialize函数，context为TradingAlgorithm的对象本身。

1. handle\_data

handle\_data会在every\_bar中调用，处理当前bar的调仓问题