Quantopian Getting Started 筆記 (Part1/2)

這邊使用官方的教程Getting Started的筆記。在這個教程中我們會先在筆記本環境中發掘一個策略，並在演算法環境中執行它。在這裡Quantopian網站有提供兩個研究環境

- 筆記本環境:在這個環境裡，我們有一個互動的python notebook環境，並可以載入quantopian提供的research模組
- 演算法環境:在這個環境裡，我們有一個寫腳本的環境，並支援zipline語法的回測框架。

所以第一步，我們會在筆記本環境裡載入數據，做簡單畫圖並確定想要執行的策略。

Step 1 如何載入資料

首先，讓我們簡單展示如何在筆記本環境載入股票資料並製作圖表。

Step 1.1 如何載入價格資料

```
# 載入資料
# 載入使用的模組
from quantopian.research import prices, symbols
import pandas as pd
# 使用quantpoian的api來呼叫蘋果股價
aapl_close = prices(
    assets=symbols("AAPL"),
    start='2013-01-01',
    end='2016-01-01',
)
#繪製圖表
# 這邊是標準的pandas api
aapl_sma20 = aapl_close.rolling(20).mean()
aapl_sma50 = aapl_close.rolling(50).mean()
# pandas 繪圖指令
pd.DataFrame({'AAPL':aapl_close,'SMA20':aapl_sma20,'SMA50':aapl_sma50}).plot(title='AAPL
Close Price/ SMA Crossover')
```

讓我們簡單講解這段程式碼，我們會把重點放在 Quantopian api的學習，pandas指令如果有問題可以在讀書會中討論(或是一些python的問題)。

首先讓我們看看 Quantopian help對這兩個函數 prices, symbols的說明:

```
prices(assets, start, end, frequency='daily', price_field='price',
symbol_reference_date=None, start_offset=0)
    Get close price data for assets between start and end.

    Parameters
    ----------
    assets : int/str/Asset or iterable of same
```

```
        Identifiers for assets to load.
        Integers are interpreted as sids. Strings are interpreted as symbols.
    start : str or pd.Timestamp
        Start date of data to load.
    end : str or pd.Timestamp
        End date of data to load.
    frequency : {'minute', 'daily'}, optional
        Frequency at which to load data. Default is 'daily'.
    price_field : {'open', 'high', 'low', 'close', 'price'}, optional
        Price field to load. 'price' produces the same data as 'close', but
        forward-fills over missing data. Default is 'price'.
    symbol_reference_date : pd.Timestamp, optional
        Date as of which to resolve strings as tickers.
        Default is the current day.
    start_offset : int, optional
        Number of periods before ``start`` to fetch.
        Default is 0. This is most often useful for calculating returns.

    Returns
    -------
    prices : pd.Series or pd.DataFrame
        Pandas object containing prices for the requested asset(s) and dates.

        Data is returned as a :class:`pd.Series` if a single asset is passed.

        Data is returned as a :class:`pd.DataFrame` if multiple assets are passed.
```

首先是prices這個函數，它接受assets, start, end, frequency, price_field, symbol_reference_date, start_offset這幾個變數，其中前三個變數是必填的，分別為要提取的資產名、提取起始時間與提取結束時間，然後回傳一個pandas dataframe物件，這邊值得注意的是在asset這邊其實也是可以直接接受字串的，但在我們的例子中先使用symbols函數轉為資產類別。

```
symbols(symbols, symbol_reference_date=None, handle_missing='log')
    Convert a string or a list of strings into Asset objects.

    Parameters
    ----------
    symbols : String or iterable of strings.
        Passed strings are interpreted as ticker symbols and resolved relative
        to the date specified by symbol_reference_date.

    symbol_reference_date : str or pd.Timestamp, optional
        String or Timestamp representing a date used to resolve symbols that
        have been held by multiple companies.  Defaults to the current time.

    handle_missing : {'raise', 'log', 'ignore'}, optional
        String specifying how to handle unmatched securities.
        Defaults to 'log'.

    Returns
    -------
    list of Asset objects
```

```
        The symbols that were requested.
```

而另一方面，symbols只是一個建構子，將字串轉為quantopian寫好的資產類別。

因此，我們的sample code只是用prices函數呼叫出apple公司的股價資料，然後使用pandas dataframe本身寫好的函數進行畫圖。

這邊我們在舉一個例子，除了price，我們可能也想直接取出回報率(return)，則我們有下列的sample code

```
# Research environment functions
from quantopian.research import returns, symbols

# Select a time range to inspect
period_start = '2014-01-01'
period_end = '2014-12-31'

# Query returns data for AAPL
# over the selected time range
aapl_returns = returns(
    assets=symbols('AAPL'),
    start=period_start,
    end=period_end,
)

# Display first 10 rows
aapl_returns.head(10)
```

Step1.2 實際使用 - 如果在回測每天自動抓取需要資料 - pipeline

但以上拉資料的例子並不能真正使用在回測中，因為在回測中，我們需要的是根據當時所能取到的資料去進行像是計算因子、分類與資料預處理等工作，這時候我們就可以利用quantopian所提供的pipeline類別，它處理我上述提到的動作，我們只需要將計算的邏輯寫好，並指定期間，pipline就可以妥善運算每次的計算。讓我們可以專心在策略的開發上面。

所以，我們真正再做拉取資料的部份都必須跟pipline配合，這也是zipline開發者希望我們採用的寫法。下面是一個簡單的例子，如何建立pipline來抓取前面提到的資料

```
from quantopian.research import run_pipeline
from quantopian.pipeline import Pipeline
from quantopian.pipeline.data import USEquityPricing

period_start = '2014-01-01'
period_end = '2014-12-31'
def make_pipeline():
    close_price = USEquityPricing.close.latest
    return Pipeline(
    columns = {

        'close_price': close_price,

    },
```

```
    )

#Pipeline execution
data_output = run_pipeline(

make_pipeline(),
    start_date=period_start,
    end_date=period_end

)
aapl_output = data_output.xs(
    symbols('AAPL'),
    level=1
)
#Plot results for AAPL
aapl_output.plot(subplots=True)
```

這邊我們先建立一個簡單的

如何載入其他官方資料集

其實在Research模組裡還有一個類別叫做Datasets，裡面有一些官方提供的其他資料集，大部份需要付費，但有一些是免費提供，我們一般的價量資料可以從上面提到的USEquityPricing這個類別中取得。而 一些基本面的資料也可以從Fundamentals這個資料集中取得。另外還有一些合作取得的資料庫，下面引用官網的文檔

Many datasets besides USEquityPricing and Morningstar fundamentals are available on Quantopian. These include corporate fundamental data, news sentiment, macroeconomic indicators, and more. All datasets are namespaced by provider under quantopian.pipeline.data.

- quantopian.pipeline.data.estimize (Estimize)
- quantopian.pipeline.data.eventVestor (EventVestor)
- quantopian.pipeline.data.psychsignal (PsychSignal)
- quantopian.pipeline.data.quandl (Quandl)
- quantopian.pipeline.data.sentdex (Sentdex)

Similar to USEquityPricing, each of these datasets have columns (BoundColumns) that can be used in pipeline computations. The columns, along with example algorithms and notebooks can be found on the Data page. The dtypes of the columns vary.

我們在這個教學將會使用到 Twitter and StockTwits Trader Mood (All fields, with Retweets) (quantopian.pipeline.data.psychsignal (PsychSignal) 這個類別)這個資料集。

首先我們模仿前面拉出幾個資料出來

```
from quantopian.research import run_pipeline
from quantopian.pipeline import Pipeline
from quantopian.pipeline.factors import Returns
from quantopian.pipeline.data.psychsignal import stocktwits

period_start = '2014-01-01'
period_end = '2014-12-31'
```

```
def make_pipeline():

    returns = Returns(window_length=2)
    sentiment = stocktwits.bull_minus_bear.latest
    msg_volume = stocktwits.total_scanned_messages.latest

    return Pipeline(
        columns = {


        'daily_returns':returns,
        'sentiment': sentiment,
        'msg_volume':msg_volume,
        }
    )

#Pipeline execution
data_output = run_pipeline(

make_pipeline(),
    start_date=period_start,
    end_date=period_end

)

#Filter results for AAPL

aapl_output = data_output.xs(
    symbols('AAPL'),
    level=1
)
#Plot results for AAPL
aapl_output.plot(subplots=True)
```