Quantitative Research Data Analytics Programming Exercise
Adam Baron – Manager of Quantitative Research Data Analytics

Portfolio Analysis Library

In this exercise you will write a software library to backtest a highly simplified portfolio trading strategy. By "backtest" we simply mean that you will implement a portfolio trading strategy and measure its performance on a historical dataset. You are given data on a subset of North American securities from 2008 and a small class library to use to access this data. Your solution should be coded in "pure" python using only the python standard library. First we will describe the data and the class library provided for its access. Part of the exercise will be to determine your ability to read the source code and understand how to use the library. Second, we will describe in simple language the nature of the portfolio simulation and list the performance measures you should calculate. You are expected to use Google to determine how to perform any calculation you are unfamiliar with.

Prerequisites

The code should work on any operating system using a standard python installation at version 2.54 or greater. Any basic laptop from within the last few years should be able to run this code with no issues. You'll need to run this code from some sort of command line environment.

The data is provided in a tgz archive. Unpack the data using the command:

$ tar xvzf qse_programming_test.tgz

Conventions

Directory names, filenames, and code entities are in **blue bold**.

Data Access Library

We use the **sqlite3** module in the python standard library to store the data you will use in this exercise. In the directory **raw_data** there are six files. You should execute the file **load_data.py** to create and populate the **sqlite3** database.

If you are familiar with **sqlite3** you can use the executable to inspect the data directly. This isn't necessary, of course, since the module **security_data.py** has a class **SecurityData** that provides an interface to access all the data you need for this exercise.

Here is a description of each table in the database:

signal

Proprietary quantitative stock ranking signals from StarMine.
security_id: unique id for a security
as_of_date: the month end date on which the data was derived
arm: StarMine analyst revisions model stock ranking score.
eq: StarMine earnings quality model stock ranking score
rv: StarMine relative valuation stock ranking score

returns

One month forward returns for North American securities in 2008.
security_id: unique id for a security
as_of_date: the month end data on which the data was derived
ret_1m: the forward one month return for the security. If you had purchased the security on this day and held it one month, you would have earned this return on your investment.

security

Identifier information for North American securities.
security_id: unique id for a security

ticker: the exchange ticker under which the security is traded. You could use this to lookup it up on google finance,
for example.
company name: the full name of the company that issued the security

sp500
The constituents of the S&P500 index in 2008.
security_id: unique id for a security
as_of_date: the date on which the index is described

security_industry
The industry assignments for each security.
security_id: unique id for a security
industry_code: the unique id for the industry

industry
The names corresponding to industry codes.
security_id: unique id for a security
industry_name: the name of the industry

You should read the code in **security_data.py** to determine how to access this data. Examples of using the library are in the file.

Portfolio Simulation

You will implement a software library for a highly simplified but realistic portfolio simulation. One could trade under the conditions of this simulation using a publically available system like FOLIOfn. In this simulation you make trades once a month on the last day of the month. You ignore transaction costs. The goal is to use the StarMine Quantitative signals or some other method (i.e. strategy) to decide which securities you want to hold at each month end date. The set of securities you decide to hold is your portfolio.
Since we are only trading on month end dates in 2008, there are exactly 12 trading days of data available to you in the database, 12/31/2007 to 11/30/2008. Thus, you will be forming 12 portfolios for each strategy.
You are to implement as many of the following strategies as you can:
1) Hold the S&P 500.
2) Hold stocks where eq>75, arm>90, and rv>50.
3) Choose any industry and hold only those stocks.
4) Hold a random 500 stocks

For each strategy you should compute as many of the following performance measures as you can:
1) The cumulative return
2) The average return
3) The *ex-post* Sharpe Ratio (use the S&P 500 as your benchmark, or use a 4% fixed rate if you are measuring against the S&P)
4) The number of “up” months
5) The number of “down” months
6) The maximum drawdown of the strategy

There is no fancy math involved in these calculations, certainly nothing more complicated than a standard deviation, and mostly just addition and multiplication.
You will need to do some data-processing on the forward 1 month returns in order for your calculations to be meaningful. This is real data, so there are outliers and nulls. You should clip returns at -0.95(-95%) and 3.0(300%).
Finally, for major bonus points, pick your favorite talking-head or columnist who picks stocks (e.g. Jim Cramer). Approximate a portfolio of their picks for each month and use your library and the data we provide to measure the performance of this “talking-head” strategy.

Implementation

You are provided with a file **trading_strategy.py** to get you started. You should include all your code in this file. You are welcome to modify or augment any of the code we have provided though the exercise was designed so that such modifications should not be necessary. We are looking for such things as correctness, idiomatic python, design aesthetics, efficiency, etc.