PROJECT

Handed out: 3 Feb 2020
Deadline 1 (Group Formation): 10 Feb 2020 (Mon), 1000 hrs
Deadline 2 (Midpoint Report): 9 Mar 2020 (Mon), 1000 hrs
Deadline 3 (Final Report): 6 April 2020 (Mon), 1000 hrs
Deadline 4 (Presentation): 13 April 2020 (Mon), in class
(NB: The blue words in this document are hyperlinks.)

In this project, you will develop a trading algorithm for futures. To expedite the development of your algorithm, we shall utilize the open-source algorithmic trading infrastructure provided by Quantiacs (www.quantiacs.com). Quantiacs is a marketplace that crowdsources trading algorithms by inviting developers to participate in competitions, investing US$500,000 - US$1,000,000 in top-performing algorithms, and allowing their developers to keep 10% of earnings with no downside risk (i.e., they do not incur any losses). Unfortunately, the competitions are only open to US residents.

1 INSTALLATION & TEST RUN

First download and install Quantiacs Toolbox by following the instructions on Quantiacs' Get-Started webpage. We shall use the Python version of the toolbox. Although the webpage claims to support Python 2.7 only, Quantiacs has announced that it supports Python 3.0 too. So the same set of instructions on the webpage will work for Python 3.0 (you will need to use `pip3` and `python3` in place of `pip` and `python`, depending on your Python setup). If you have *both* Python 2.7 and Python 3.0 installed on your machine, then *may the heavens have mercy on you.* The installation should still go through as long as you consistently use one version. The Quantiacs' open-source code is located in the `quantiacs-python` directory. After downloading the sample code (`trendFollowing.py`) from github as mentioned on the webpage, you can conduct a test run of the installation in a terminal with the command `python trendFollowing.py` (rather than `python path/to/trading_system.py` as stated on the webpage). When you run the code for the first time, it will download 30 years' worth of futures data (1 Jan 1990 till the current date), which could take a long time. To speed up the test-run, you may shorten the `settings['markets']` list in `trendFollowing.py`. After you have downloaded the futures data (stored in the `tickerData` directory), the execution of the code occurs locally on your machine. By default, the data will not be downloaded again, so subsequent runs will not incur the download overhead.

2 TOOLBOX DOCUMENTATION

Read through the Python documentation of the toolbox. Next, scan through `quantiacsToolbox/quantiacsToolbox.py` to have an overview of the code's flow. You may wish to put breakpoints at the start of the functions `myTradingSystem` and `mySettings` in `trendFollowing.py` to see how many times they are called, and where they fit in the overall flow of the code. You may also wish to read the step-by-step explanation of `trendFollowing.py`, and a high-level description of the components of an algorithmic trading system.

3 DATA

For this project, you will be using the price data of 88 futures. Although Quantiacs also provides data on stocks, the project limits you to trading futures only because of computational considerations (there are over 480 stocks, a large number that may burn a hole in your laptop if you use sophisticated models that require intensive computation). Quantiacs has abstracted away the complexities of futures (see the section on future roll-overs in the toolbox's documentation above), so you can trade them as if they are stocks. If you would like to find out more about futures, see this Quantiacs video.

A text file containing the full list of futures is available here. You should set `settings['markets']` to this list and add `'CASH'` to it. `'CASH'` is the "financial instrument" to which you can apportion the fraction of your budget that you do not wish to invest in any future. (This should be clearer after you have read through the toolbox's documentation.)

You can also use these macro-economic indicators in your trading algorithms.

4 RESOURCES

You may consult Quantiacs' videos, presentations, FAQs and sample code, as well as, other external resources for inspiration on the form of your trading model. (Please do not copy code from your references!)

5 PROJECT SPECIFICATION

5.1 Function Definition

You are to define the functions `myTradingSystem` and `mySettings`. The former contains your algorithm for daily trades, and the latter contains parameter specifications for your trading algorithm. See this webpage of the toolbox documentation for details. You can run your trading algorithm by calling the `quantiacsToolbox.runts` function. This function backtests the trading algorithm in `myTradingSystem` and returns several important pieces of information about your algorithm. Among these is Sharpe ratio, which is the ratio of average return to average volatility (a typical investor would want a large return with low volatility; see Section 5.2 for details). You need not restrict your algorithms/models to those that you learn in class. You are free to utilize the full span of your knowledge to develop a trading algorithm/model that gives stellar results. You may wish to first develop your algorithm/model outside of the Quantiacs' framework, and then add it to `myTradingSystem` for backtesting.

5.2 Sharpe Ratio

The Sharpe ratio is calculated as follows. Let $\{e_j\}_{j=0}^n$ be the set of future values of a portfolio over a period of $n+1$ trading days. A vector of daily returns $\{d_j\}_{j=1}^n$ is given by $d_j = \frac{e_j - e_{j-1}}{e_{j-1}}$.

The average daily return is given by $r_{daily} = \left(\prod_{j=1}^n 1 + d_j\right)^{1/n} - 1 = \left(\frac{e_n}{e_0}\right)^{1/n} - 1$.

The yearly return is given by $r_{yearly} = (1 + r_{daily})^{252} - 1$. (There are 252 tradings days in a year.)

Daily volatility is computed as $vola_{daily} = standardDeviation(\{d_1, d_2, \ldots, d_n\})$.

Yearly volatility is computed as $vola_{yearly} = \sqrt{252}\, vola_{daily}$.

Finally, the Sharpe ratio is defined as $Sharpe = \frac{r_{yearly}}{vola_{yearly}}$. (NB: The higher the Sharpe ratio, the better the performance of the portfolio.) You may wish to peruse the Python code for the Sharpe ratio in the `stats` function in `quantiacsToolbox.py` to see how Quantiacs compute it. (NB: An advantage of using open-source code is that you can freely dig into how the code works.)

5.3 Training and Test Data

As mentioned in Section 1, the `quantiacsToolbox.runts` function downloads 30 years' worth of futures data (1 Jan 1990 till the current date) into the `tickerData` directory when it is first executed.

You can use all the data from 1 Jan 1990 till 31 Dec 2019 (or some subset of it) for developing, training and tuning your trading algorithm. After developing your algorithm, you have to backtest it on the test data from 1 Jan 2020 to 31 Mar 2020. The test data must **never** be used to train or tune your algorithm. Your algorithm is to be evaluated according to its Sharpe ratio over the period of the test data. The test data will have to be downloaded as they become available (either delete the `tickerData` directory and execute `quantiacsToolbox.runts`, or call `quantiacsToolbox.loadData`).

5.4 Settings

In the `mySettings` function, set your budget to $1 milllion (`settings['budget']= 10**6`), limit your `myTradingSystem` function to accept data from the past 504 days (`settings['lookback']= 504`; this is orthogonal to using the training data to train your model), and set slippage to 0.05 (`settings['slippage']= 0.05`). Further, please set `settings['beginInSample']` and `settings['endInSample']` to appropriate dates, and `settings['markets']` to the full set of futures plus `'CASH'`.

5.5 Deliverables & Deadlines

(a) You are work on this project in a group of 3 or 4 (there are to be seven groups of 4, and two groups of 3). Please form your group and email Joel (e0210482@u.nus.edu) your group name and members by **10 Feb 2020 (1000 hrs)**. (All members of a group are expected to participate actively in the project. If any member decides to be a *passenger*, please let Stanley know early.)

(b) Email Joel a one-page report describing your progress (your proposed model, the models you have experimented with, their performances, next steps, etc.) by **9 Mar 2020 (1000 hrs)**.

(c) Email Joel your code and final report by **6 Apr 2020 (1000 hrs)**. The report should include, but is not limited to:

- A detailed exposition of your trading algorithm/model. You have to demonstrate that you understand how your model works, be able to explain why it works well (or not), and explain you how arrive at your model. Please do not simply say, "We threw everything but the kitchen sink at the problem, and model X works best...but I don't know why."
- Different approaches that you have tried (if any), and their respective performances.
- A description of your manipulations of the data (if any).
- A detailed description of how you train and tune your model on the training set. (You should **never** train or tune your model on the test set.) A step-by-step guide can be included as an appendix. This allows us to replicate how you obtained your final model. (You will be penalized if your model cannot be replicated.)

- Sharpe ratios on the training data and test data. (You will be penalized if these numbers cannot be replicated when we run your code.) The performance of your model will be gauged by its Sharpe ratio over the test data. We reserve the right to test your algorithm on additional data beyond the test dates to ascertain its performance.

The final report should not exceed 5 pages (Times 12 pt font).

(d) Give a 15-min presentation of your final project in class on **13 April 2020**.

5.6 Grade Breakdown

This project is worth 20% of your final BT4013 grade. The grade breakdown for the project is shown below. We reserve the right to exercise discretion on how points are allocated, especially for the "Performance on Test Data" component.

Item	Weightage
Timely Group Formation	5
Midpoint report	5
Final Report	40
Presentation	20
Performance on Test Data	30

Table 1: Project grade breakdown