Linear Programming Backtest Toolbox V1.0

User Guide

2011-06-28

Table of Contents

1 Introduction 3

2 Function Help 4

2.1 Functions to be explicitly called by users 4

LoadData 4

LoadParameter 5

Main_Backtest 6

2.2 Functions not to be explicitly called by users 7

LoadWorkSpace 7

ConstraintBuider 7

PFConstruction 7

RefineAssetConstraint 8

LinearOptimize 8

PFAnalytics 8

PFReport 8

3 An Example: Step by Step Guide on Running a Backtest 9

Step 0 – Add path 9

Step 1 – Prepare your alpha (signal) 9

Step 2 – Load the input data 9

Step 3 – Load the parameter 10

Step 4 – Run backtest 10

Step 5 – Evaluating the backtest result 10

4 FAQ 11

Appendix A - Optimization Problem in Mathematical Form 12

A.1 Optimization problem – original 12

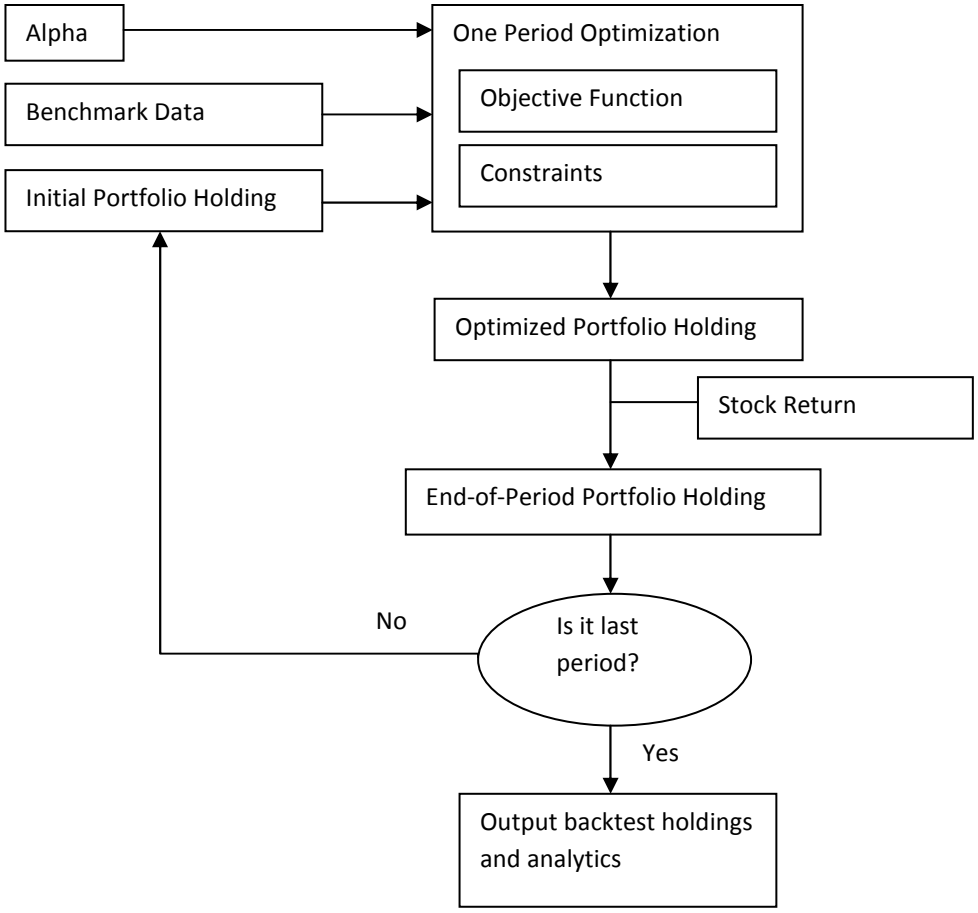
A.2 Optimization problem - with alpha pick-up 13

1 Introduction

This backtest toolbox provides users with a complete strategy backtest / portfolio constructon process. It can be used to:

1) Load necessary data for backtest from database
2) Customize parameters for constructing optimal portfolio
3) Customize both asset level / attribute level constraints for constructing optimal portfolio (including maximum stock bet, sector / country neutral, liquidity constraints, etc)
4) Perform multi-period portfolio optimization and backtest within a reasonable time frame
5) Calculate key analytics for optimized portfolio and output the backtest result into a PDF.

Graph below presents a flow chart of the basic backtest procedures



Portfolio optimization in this toolbox is based on linear programming. Non-linear constraints / objectives cannot be dealt with in this version. Risk model is not required as an input.

2 Function Help

This toolbox and its underlying matlab functions can be found on Team Foundation Server (TFS) under:

$/QuantStrategy/Analytics/Utility/LinearOptimize/Matlab

You need get latest of this folder before you can use this toolbox.

2.1 Functions to be explicitly called by users

There are only three functions which have to be explicitly called by users - LoadData, LoadParameter, and Main_Backtest.

LoadData

Load necessary data for backtest and save them as a .MAT file

Syntax

```
LoadData(filename, signal, aggid, startdate, enddate)

LoadData(filename, signal, aggid, startdate, enddate, freq)

LoadData(filename, signal, aggid, startdate, enddate, freq, startpf)
```

Input

filename – <string> the name of the .MAT file to be saved to

signal – if a myfints, it contains stock signals (alphas), if a string, it is factorId as defined in quantstrategy.fac.factormstr. Factor with this id will be used as alpha

aggid – <string> the id of benchmark as defined in quantstaging.dbo.aggmstr

startdate – <string> the starting date of backtest

enddate – <string> the end date of backtest

freq – (optional) <string> the frequency of backtest, can be one of the following:

'M' – monthly

'A' – annually

'Q' – quarterly

'W' - weekly

Default value = 'M'.

startpf – (optional) <myfints> the initial portfolio holdings at the start of the backtest.

Default value = empty, meaning starting from cash

Output

filename.MAT - will be saved in the directory, and it stores following variables:

bmhd - <myfints> benchmark holding in weight

signal - <myfints> signal (alpha)

fwdret - <myfints> one period forward total return (length of period depends on the freq)

gics - <myfints> historical GICS

price - <myfints> adjusted price of stocks in USD

ctry - <cell array> static country attribute of stocks as defined in quantstaging.dbo.secmstr

freq - <string> the frequency of all myfints above, same as the input freq of backtest

adv - <myfints> trailing one month average daily dollar volume

startpf - <myfints> the starting portfolio, same as input but has been aligned with other myfints

Note: You can also construct your own .MAT file as long as it has the same variable names and data types as above, and all myfints are aligned. Among variables above adv and startpf are optional.

LoadParameter

Load backtest parameters and return a structure

Syntax

```
parameter = LoadParameter

parameter = LoadParameter('pickup',0.2,'actbet',0.01,...)

parameter = LoadParameter('param1',value1,'param2',value2,...)
```

Input

param1 , param2 … - (optional) <string> the parameter name, can be one of the following:

'pickup' – the alpha pickup used to control turnover (higher pickup, less turnover), refer to appendix A.2 for more information on pickup technique

'actbet' – the maximum active bet (on both side) for each stock in the benchmark

'percentlq' – the maximum percent of ADV can be traded for each stock

'capital' – the initial dollar amount of capital of portfolio

'tcost' – the transaction cost as a percentage of per unit of turnover

'sectorbet' – the maximum sector bet for each sector (= 0 means strict sector neutrality)

'ctrybet' – the maximum country bet for each country (= 0 means strict country neutrality)

'propactbet' – used to adjust maximum active bet of each stock to be proportional to values of an attribute, with the value assigned to 'actbet' as the maximum.

(..'propactbet', 'signal',…) – max bet proportional to the abs. value of signal

(..'propactbet', 'bmhd',…) – max bet proportional to the benchmark weight

Value1, value2… - (optional) the parameter value. Following table gives data type and default value of each parameter

Parameter	Data Type	Default Value	Explanation
pickup	Numeric	0.2	A 0.2 alpha pickup is enforced
actbet	Numeric	0.005	Max active bet of each stock is 50 bps
percentlq	Numeric	0.2	Trade size of a stock is at most 20% of its ADV
capital	Numeric	100,000,000	Portfolio starts at 100 million USD
tcost	Numeric	0	Transaction cost is 0
sectorbet	Numeric	0.01	Max active bet on sector is 1%
ctrybet	Numeric	Inf	Max active bet on country is unlimited
propactbet	String	''	Max active bet is not proportional to anything

Output

parameter - < structure > a structure containing the fields listed in the table above.

Main_Backtest

Main function of the backtest

Syntax

```
[portfolio, analytics] = Main_Backtest(datafile, parameter, result2file)
```

Input

datafile - <string> the name of the .MAT file stores the input data for backtest (from **LoadData**)

parameter - <structure> the parameter structure from **LoadParamter**

result2file - <logical>

0: not store result into any file.

1: store result into .MAT file with the name 'datafile_result.MAT' and backtest report into .PDF with the name 'datafile_report.PDF'

Output

portfolio - < structure > contains optimal portfolio with following fields

portfolio .opthd – <myfints> optimized portfolio holding in weights

portfolio .inihd – <myfints> initial portfolio holding in weights

portfolio.optshr - <myfints> optimized portfolio holding in shares

portfolio.tradeshr - <myfints> trading size in shares

a**nalytics** - < structure> contains key analytics with following fields

analytics.actret - <myfints> active return after cost

analytics.cumret - <myfints> cumulative active return after cost

analytics.TO - <myfints> turnover

analytics.pfcost - <myfints> portfolio level transaction cost

analytics.IC - <myfints> information coefficient of alpha

analytics.TC - <myfints> transfer coefficient between alpha and active weight

analytics.signalexp - <myfints> active alpha exposure of portfolio

analytics.nlong - <myfints> number of long names in portfolio

analytics.ntrade - <myfints> number of trades

analytics.autocorr - <myfints> auto correlation of alpha

analytics.drawdown - <myfints> active drawdown

analytics.yearlyreturn - <myfints> yearly active return

datafile_result.MAT - .MAT file storing structures: portfolio and analytics

datafile_report.PDF - .PDF file storing backtest report

2.2 Functions not to be explicitly called by users

There are 7 utility functions which are unlikely called by users explicitly but will be called internally by the main function. A brief introduction is given below:

LoadWorkSpace

Load the required variables from the .mat file and the parameter structure, perform necessary check on the data, and generate constraints according to inputs

ConstraintBuider

Return constraints in cell arrays with standard data structure according to the input. It works for both asset level and attributes level constraints.

PFConstruction

Construct multi-period portfolios using linear programming optimization

RefineAssetConstraint

Combine the individual asset constraint to return the final lower bound, upper bound of position, and maximum trade size of buy / sell

LinearOptimize

linear optimization for portfolio construction (one period)

PFAnalytics

Calculate key analytics for the constructed portfolio including active return, turnover, transfer coefficient, drawdown, IR, IC, tracking error, etc.

PFReport

Take the output from PFAnalytics and present the backtest result in a report with graphs and statistics. It saves the report into a PDF.

3 An Example: Step by Step Guide on Running a Backtest

It is very easy to run a backtest using the toolbox. This session will give a step-by-step illustration.

Step 0 – Add path

You need to add the correct path in matlab before you can start the backtest. This toolbox is under:

'\QuantStrategy\Analytics\Utility\LinearOptimize\Matlab\'

Other necessary paths are:

'\QuantStrategy\Analytics\Utility\'

'\QuantStrategy\Analytics\Utility\myfintsUtility\'

'\QuantStrategy\Analytics\Utility\myfintsUtility\UserDefinedFun\'

'\QuantStrategy\Analytics\Utility\Unverified\'

'\QuantStrategy\Analytics\FactorLib\dev\'

'\FinancialUtility\runSP\'

'\FinancialUtility\bulkInsert\'

Step 1 – Prepare your alpha (signal)

You need to prepare the alpha you intend to backtest in a myfints, the fields of the myfints should be valid secid of stocks. If you are not familiar with myfints, please refer to the user guide of factor toolbox for more information.

If you don't have any alpha but just want to try out this backtest tool, pick whatever factorId you like, say, 'F00001' (book value to price), and you can pass this id to the LoadData function as the signal (You can go to **quantstrategy.fac.factormstr** in DB to find id and definition of factors).

Please note that if you pass in your own alpha as a myfints, the backtest tool will **NOT** normalize it for you, so please make sure the alpha you prepared is already normalized to your need.

Step 2 – Load the input data

In the example below, we load data for backtest using factor 'F00001' as the alpha, and SP500 as the benchmark. The backtest time frame is from Jan 2000 to May 2011, and the rebalancing frequency is monthly.

```
% input
filename = 'DEMO';
```

```
signal = 'F00001'; % note signal here can also be a myfints prepared by user
aggid = '00053';
startdate = '2000-01-01';
enddate = '2011-05-31';
freq = 'M';

% load data
LoadData('DEMO', 'F00001', '00053', '2000-01-01', '2011-05-31', 'M');
```

After execute the function, we should find a .MAT file with the name 'DEMO' in the directory.

Step 3 – Load the parameter

Continue with the previous example, we now customize the backtest parameter:

```
parameter = LoadParameter('pickup',0.2,'actbet',0.01,'sectorbet',0);
```

Here we set the pickup to be 0.2 and max active bet to be 1%. We also want the portfolio to be strictly sector neutral so we set sector bet equal to 0.

If you are lazy, you can just call the function without any input, it will generate a parameter structure with default values:

```
parameter = LoadParameter;
```

You can change the field value in this structure later if you find the default parameter values don't give you satisfactory result.

Step 4 – Run backtest

Continue with the previous example, we call the main function to run the backtest and store the result into files.

```
[portfolio, analytics] = Main_Backtest('DEMO',parameter,1);
```

During the process we will see messages indicating the optimization status, please be cautious about the result if you see any warnings printed on the screen during the optimization process.

After executing the function, we will find the DEMO_result.MAT and DEMO_report.PDF appear in the directory.

Step 5 – Evaluating the backtest result

Open the report pdf to look at the graphs and statistics of active return, turnover, TC, IC, alpha exposure, number of holding, activeness, and alpha auto correlation.

For more details of the portfolio and backtest analytics, you can load the DEMO_result.MAT file into workspace, and check the data inside.

4 FAQ

1) ***Is the portfolio long-only and full-invested in the backtest? Can it invest in stocks outside the benchmark?***

   The backtest tool has several basic assumptions on the portfolio construction, among which there are *'long-only'* (weight of stock >= 0), *'full investment'* (sum of total weight = 100%), and *'Restrict to BM'* (only invest in stocks in the benchmark). This version doesn't support long-short portfolio construction.

2) ***How to control tracking error when there is no risk model?***

   Although there is no risk model appearing in optimization, you can control tracking error by adjusting maximum stock bet, sector bet, and country bet.

3) ***How to control turnover?***

   Turnover can be controlled by adjusting the alpha pickup. Increasing the pickup will make the trade harder to make, and hence reduce the turnover. Reducing maximum stock bet can also decrease the turnover.

4) ***How to control rebalance frequency? Can the toolbox do irregular rebalancing?***

   You can control rebalance frequency by inputting the desired parameter *freq* when loading the data, it currently support annually, quarterly, monthly, and weekly rebalancing. For instance, the command below will set the frequency to be quarterly.

   ```
   LoadData(filename, signal, aggid, startdate, enddate, 'Q')
   ```

   The current version doesn't support irregular rebalancing date.

5) ***What will happen if the constraints contradict to each other?***

   There are at least two cases of inconsistent constraints:

   - Case 1: Inconsistent asset level constraint. In this case, constraints end up with inconsistent lower bound and upper bound for stocks (meaning the lower bound is higher than the upper bound). The optimizer will proceed by enforcing those lower bounds to equate the upper bounds, and print out a warning message on the screen to indicate number of inconsistent observations. This case is most likely caused by trading liquidity constraints.
   - Case 2: Inconsistent attribute level constraint. In this case, inconsistent constraints lead to infeasible solution or non-optimal solution. Optimizer will print out a warning message on the screen indicating the failure of optimization. The result obtained cannot be trusted. This case is most likely caused by strict sector / country neutral constraints.

Appendix A - Optimization Problem in Mathematical Form

This session presents the basic math for a single period optimization.

A.1 Optimization problem – original

Objective:

$$Maximize \sum_{i}^{N} w_i \alpha_i$$

Asset Level Constraints:

$$w_i^{lb} \leq w_i \leq w_i^{ub}$$

Attribute Level Constraints:

$$\sum_{i}^{N} w_i = 100\%$$

$$\sum_{i}^{N} A_i^j w_i = B^j \, , j = 1,2,3 \ldots$$

$$B_{lower}^k \leq \sum_{i}^{N} A_i^k w_i \leq B_{upper}^k, k = 1,2,3 \ldots$$

w_i is the weight of stock i in the portfolio

α_i is the alpha of stock i

N is the total number of stocks in the universe

w_i^{lb} is the lower bound of the weight of stock i in the benchmark

w_i^{ub} is the upper bound of the weight of stock i in the benchmark

A_i^j is the value of attribute j for stock i

B^j is the value used to force the equality constraint for attribute j

B_{upper}^k and B_{lower}^k is the upper / lower boundary values to force the inequality constraint for attribute k

A.2 Optimization problem - with alpha pick-up

To control the turnover in portfolio construction, an alpha pick-up technique is implemented in the optimization: a stock will not be purchased / sold unless the resulting increment of alpha exceeds a threshold (we call it alpha pickup), given all constraints are satisfied.

To implement this technique, the optimal portfolio (weight denoted by *w*) is decomposed into two portfolios:

1) Portfolio of all new buys (weight denoted by $\widetilde{w}$)
2) Portfolio of existing position with new sells (weight denoted by $\overline{w}$)

We have $w = \widetilde{w} + \overline{w}$ (optimal portfolio = 'New buy' portfolio + 'Hold or sell' portfolio)

In this way, the optimization problem is transformed to the following format:

Objective:

$$Maximize \sum_{i}^{N} \overline{w}_i(\alpha_i + p) + \sum_{i}^{N} \widetilde{w}_i \alpha_i$$

Asset Level Constraints:

$$\min\left(w_i^{PF}, w_i^{lb}\right) \le \overline{w}_i \le \min\left(w_i^{PF}, w_i^{ub}\right)$$

$$\max\left(w_i^{lb} - w_i^{PF}, 0\right) \le \widetilde{w}_i \le \max\left(w_i^{ub} - w_i^{PF}, 0\right)$$

Attribute Level Constraints:

$$\sum_{i}^{N} \overline{w}_i + \widetilde{w}_i = 100\%$$

$$\sum_{i}^{N} A_i^j(\overline{w}_i + \widetilde{w}_i) = B^j, j = 1,2,3 \ldots$$

$$B_{lower}^k \le \sum_{i}^{N} A_i^k(\overline{w}_i + \widetilde{w}_i) \le B_{upper}^k, k = 1,2,3 \ldots$$

Other than notations that have appeared in A.1, we also have:

p is the pick-up value for alpha

$\overline{w}_i$ is the weight of stock i in the 'Hold or Sell' portfolio

$\widetilde{w}_i$ is the weight of stock i in the 'New Buy' portfolio

w_i^{PF} is the weight of stock i in the initial portfolio (or end-of-last-period portfolio)