Factor Performance Timings

2019-10-28

My Data Outlet

Introduction

This post is intended to give readers a better understanding of the speed of the MDO data retrieval functions. The MDO Platform provides an API for users to access pricing, estimates, and fundamental data. MDO also provides functionality to easily create and test quantitative factors. We believe that the ability to run backtests in a reasonable amount of time (minutes) is critical to the research process. Every data call and calculation is tested and optimized for performance.

These examples will show some typical processing times for calculating commonly used factor formulas. For our timing, we use a universe of around 3,000 securities on month end dates from 6/30/2005 to 9/30/2018.

Summary of Results

We calculated Factor Values and Performance Statistics for around 3,000 securities on each month end date from 6/30/2005 to 9/30/2018. This totals 160 dates (> 470,000 total observations). Total backtest times were:

- **Single Factor: 70 seconds**
- **Multi Factor: 3 minutes**
- **Multi Factor (custom data): 35 seconds**

Single Factor Timings

For this demonstration, we will run a generic Return on Equity (`ROE`) factor. Let's start by retrieving the constituents of the Russell 3000 index.

```
day <- DateRange(startDate = 20050630, endDate = 20180930, periodType = 'month', dom = 31)
universe <- ConstituentUniverse(items = "DS_CONST_MTH__FRUSSL3", days = day)
```

Next, we will run the `QFPerformance()` function on our factor, which will first calculate raw factor values. Then, QFPerformance() will calculate information coefficients and quantile peformance over various time horizons. The `n = 5` argument indicates that you want to quintile the factor.

This will take around **70 seconds** in total.

```
# Run quant factor performance
factors <- c('MDQ_ROE')
performance <- QFPerformance(universe, factors, n = 5, returnHorizon = list(horizon = 12, pe
```

Multi-Factor Timings

The MDO platform also allows users to backtest *multiple* factors at a time, as well as models or submodels. To calculate performance for the 7 below factors (plus 1 random factor), this takes around **3 minutes** in total.

```
# Run quant factor performance
factors <- c("MDQ_ROE", "MDQ_SURP_SUE", "MDQ_RANDOM", "MDQ_ASSET_GROWTH_5Y",
             "MDQ_BUYBACK_YLD", "MDQ_GROSS_MARGIN", "MDQ_PRICE_SALES", "MDQ_ROA")
performance <- QFPerformance(universe, factors, n = 5, returnHorizon = list(horizon = 12, pe
                             sortOrder = QFInfo(factors, 'sortOrder')) # 180 seconds
```

back to top

Multi-Factor Timings (using custom data)

For even faster backtesting performance, you can load historical factor values into custom data tables.

In the below example, we again calculate historical factor values. We then create a custom data table and load those values. This will take around 2.5 minutes to calculate values and an additional 11 seconds to create and load the data into a SQL table.

```
CustomDataLoad(name = 'MDO_FACTOR_VALUES', x = universe, userGroup = 'research') # 11 seconds
```

Retrieving the raw data will take around 1 second to run, using our same historical universe.

```
CustomData(universe, items = customFactors, recent = TRUE, update = T, userGroup = 'research
```

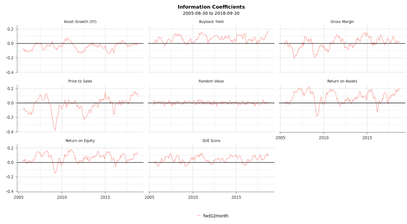
Calculating QFPerformance() on the *pre-calculated* values will reduce the total time from 3 minutes to around **35 seconds**. By retrieving the data from a custom data table, we can eliminate the time spent on calculating our factors on the fly. This is also useful for more granular (*weekly, daily*) backtesting.

```
performance <- QFPerformance(universe, customFactors, n = 5, returnHorizon = list(horizon =
                             sortOrder = QFInfo(factors, 'sortOrder')) # 35.25 seconds
```

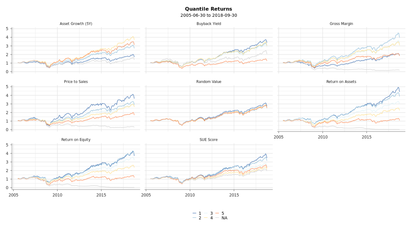
Factor Performance Results

Whether we calculate data on the fly or retrieve the data from custom data, we can return the same backtest statistics and analysis to inform our decisions about these factors. The plots below show a few of the backtest statistics available in `QFPerformance()` .

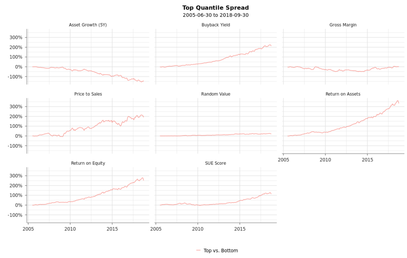
```
# Plot the information coefficients for each horizon
SummaryData(performance, "ICDetail", type = "plot")
```




```
# Performance of each quantile
SummaryData(performance, "QuantileReturns", type = "plot")
```




```
# Performance of top quantile vs. bottom quantile
SummaryData(performance, "TopQuantile", type = "plot", versus = "Bottom")
```



Appendix

SQL Server Specs

- Operating system: Windows (Windows Server 2016 Datacenter)
- Size: Standard DS14 v2 (16 vcpus, 112 GiB memory)
- Disk: Azure Premium SSD (5000 IOPS limit/200 MB/s)

Factor Methodology

- 'MDQ' is the factor naming prefix used in the generic mdo.factors library which is provided to all clients.
- Fundamental factors use an avail of Quarterly, Semi-Annual, and Annual data.
- Fundamental factors use Quarterly TTM for Income Statement and Cash Flow items.

back to top