Historical Stock Data

While there are plenty of data solutions aimed at professional money managers and traders, many of these tend to be too expensive for the average investor. Some semi-professional trading platforms such as Interactive Brokers offer data retrieval capabilities via an api, but the quantity and frequency of data is often quite limited. For example, IB limits data access to 100 simultaneous data requests (the use can pay extra for more), while often generating a “pacing violation” error if the update requests sent via the api become too frequent.

Probably the most reliable source of historical and real-time data is Yahoo Finance and there are many methods one can use to access the data, whether via Excel, Matlab or, as we explore in this post, Mathematica.

Retrieving Stock Data Using Mathematica

Mathematica offers a link to the Yahoo Finance data via its FinancialData function (see http://reference.-wolfram.com/language/ref/FinancialData.html). The function is able to access a great many of the data fields available via the Yahoo api, but by no means all of them: amongst the important gaps, for instance, is the upcoming earnings release date (or, rather, the date range, where the actual release date is uncertain). An even larger omission is options data, available on the Yahoo Finance site, but not via FinancialData, or via Matlab’s equivalent Fetch function from the Datafeed Toolbox. There are several user-supplied solutions to address this problem, both in MatLab and Mathematica, which can be found either on the Mathworks File Exchange site (for Matlab) or on the Mathematica Stackexchange site. Unfortunately, Yahoo reconfigures its api quite often, causing these homespun solutions to fail. At the time of writing I am unaware of any available code base in Matlab or Mathematica that is able to access options data from Yahoo Finance successfully.

However, when it comes to stock-related data, Mathematica’s FinancialData function makes the retrieval process very straightforward. In this example we begin with a list of stock tickers we want to access data for:

```
nStocks = Length[tickers = Flatten[Import["/Users/.../tickers.csv"]]]
```

```
886
```

As mentioned earlier, there is a very wide range of data that can be retrieved using the FinancialData function:

```
FinancialData["Properties"]
```

```
{Ask, AskSize, Average200Day, Average50Day, AverageVolume3Month, Bid, BidSize,
 BookValuePerShare, Change, Change200Day, Change50Day, ChangeHigh52Week,
 ChangeLow52Week, CIK, Close, Company, CumulativeFractionalChange, CumulativeReturn,
 CUSIP, Dividend, DividendPerShare, DividendYield, EarningsPerShare, EBITDA,
 Exchange, FloatShares, ForwardEarnings, ForwardPERatio, FractionalChange,
 FractionalChange200Day, FractionalChange50Day, FractionalChangeHigh52Week,
 FractionalChangeLow52Week, High, High52Week, ISIN, LastTradeSize, LatestTrade,
 Lookup, Low, Low52Week, MarketCap, Name, OHLC, OHLCV, Open, PEGRatio, PERatio,
 Price, PriceTarget, PriceToBookRatio, PriceToSalesRatio, QuarterForwardEarnings,
 Range, Range52Week, RawClose, RawHigh, RawLow, RawOHLC, RawOpen, RawRange, Return,
 Sector, SEDOL, ShortRatio, SICCode, StandardName, Symbol, Volatility20Day,
 Volatility50Day, Volume, Website, YearEarningsEstimate, YearPERatioEstimate}
```

In many cases (although by no means all) we are interested in stock returns for analytical purposes and FinancialData enables us to retrieve returns directly rather than compute them from raw stock prices. In this example we download daily returns over the last year for our list of 886 symbols, a process takes only a few minutes.

```
rawdata = FinancialData[#, "Return",
      {DatePlus[Today, {-1, "Year"}], DatePlus[Today, {-1, "Day"}]}] & /@ tickers;
```

If we could safely assume that data exists for all the selected dates for every stock in our chosen universe we could proceed directly to the next stage and the analysis would be made more efficient by using vectorized functions. Unfortunately, this is typically not the case - data will often be missing or invalid for some members of the universe. In this example, several stocks are missing returns data for one or more days during the last year:

```
tickers[[
 Flatten[Position[Table[Length[rawdata[[i]]], {i, 1, nStocks}], x_ /; x < 252]]]]
```

```
{ALK, BPL, CPHD, DD, IOSP, MNST, MSM, TCO, TFX, XEC}
```

So, in what follows, we have to use a more pedestrian approach compared to the more elegant solution we first attempted, making full use of the matrix manipulation capabilities of the language.

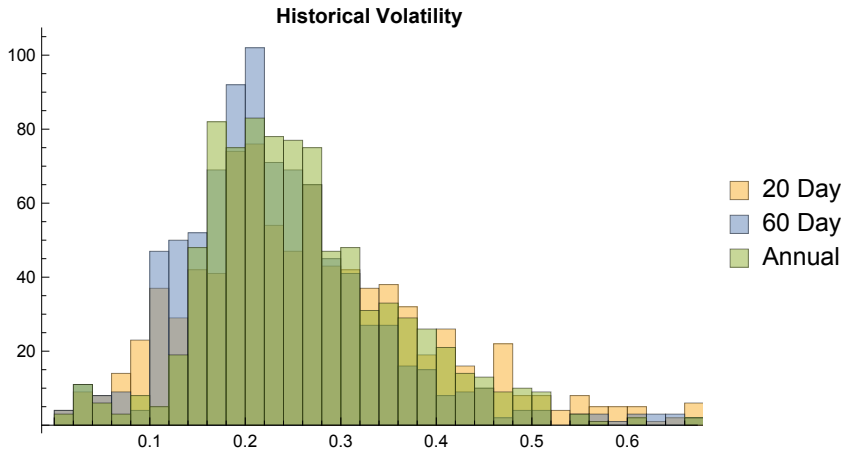
Data Analysis

The great advantage of using a tool like Mathematica for data collection is that one can immediately apply the wealth of functionality available within the software for data preparation and analysis. Here, for example, we are interested in exploring the changes in historical volatility in the stock universe over time, as an input to a volatility arbitrage strategy. We estimate the annualized volatility over the preceding month (20 days) and 3-month period (60 days) and compare these estimates to the estimated annual volatility for each stock. Our ex-ante expectation is there are likely to be significant differences in the volatility distributions which may be due in part to changes in the volatility of the asset processes during earnings season.

```
Dimensions[historicalVolatility =
  Transpose[Table[Sqrt[Length[rawdata[[1]]]] * Table[StandardDeviation[
       Take[rawdata[[i]][[All, 2]], l]], {l, {-20, -60, All}}], {i, 1, nStocks}]]]
```

```
{3, 886}
```

```
Histogram[historicalVolatility, PlotLabel → Style["Historical Volatility", Bold],
 ChartLegends → {"20 Day", "60 Day", "Annual"}]
```



The histogram plots indicate the possibility of significant differences in the cross-sectional means and also possibly the higher moments also of the volatility distributions. Let's take a closer look:

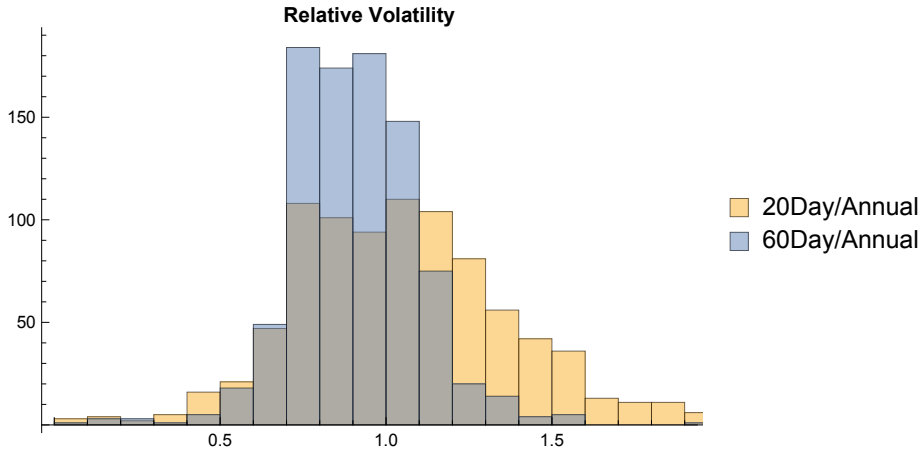
```
headings =
 {{"Min", "Max", "Median", "Mean", "Standard Deviation", "Skewness", "Kurtosis"},
  {"20 Day", "60 Day", "Annual"}};
TableForm[Transpose[{Min[#], Max[#], Median[#], Mean[#], StandardDeviation[#],
     Skewness[#], Kurtosis[#]} & /@ historicalVolatility], TableHeadings → headings]
```

	20 Day	60 Day	Annual
Min	0.00527633	0.00522165	0.0057319
Max	2.86684	1.65288	1.37033
Median	0.252289	0.219629	0.245624
Mean	0.284375	0.24093	0.269294
Standard Deviation	0.180096	0.12619	0.133002
Skewness	4.40476	3.14631	2.67176
Kurtosis	52.3335	26.0951	17.6399

The pattern appears to be that volatility in the 3 months leading up to the earnings seasons has been significantly lower on average than over the preceding year, but has risen in recent weeks to a level that exceeds the average annual volatility. It is perhaps unremarkable that stocks become more volatile during earnings season and that, furthermore, the distribution of 20-day volatility has much higher dispersion, skewness and kurtosis than the distribution of annual volatility, presumably due to the fact that some stocks are likely to make substantial moves after an earnings announcement.

Another way of envisaging the effect is to look at the relative volatility during the prior 1-month and 3-month periods, compared to the annual volatility:

```
Histogram[relativeVolatility =
  {historicalVolatility[[1, All]] / historicalVolatility[[3, All]],
   historicalVolatility[[2, All]] / historicalVolatility[[3, All]]},
 PlotLabel → Style["Relative Volatility", Bold],
 ChartLegends → {"20Day/Annual", "60Day/Annual"}]
```




```
headings = {{"Min", "Max", "Median", "Mean",
   "Standard Deviation", "Skewness", "Kurtosis"}, {"RV20/252", "R60/252"}} ;
TableForm[Transpose[{Min[#], Max[#], Median[#], Mean[#], StandardDeviation[#],
     Skewness[#], Kurtosis[#]} & /@ relativeVolatility], TableHeadings → headings ]
```

	RV20/252	R60/252
Min	0.0314348	0.0639674
Max	3.4401	1.9834
Median	1.03732	0.901882
Mean	1.06717	0.908993
Standard Deviation	0.362406	0.191582
Skewness	0.932216	0.173124
Kurtosis	6.13597	5.47203

So, amongst the universe of stocks considered here, volatility in the most recent month is around 7% higher during the most recent than the historical annual average, while volatility during the prior 3-month period has been approximately 9% lower than the annual average.

Options Data

The next stage in the development of the volatility strategy might be to gather data on option contracts for our universe of stocks. There are several commercial sources of options data available but, again, Yahoo Finance offers the investor a source of free data of reasonable quality. Yahoo Finance is superior to Google Finance in this regard, since the latter doesn't appear to offer data for weekly options, which many investors like to trade.

Apple Inc. (AAPL) ☆ Add to watchlist

NasdaqGS - NasdaqGS Real Time Price. Currency in USD

108.43 +0.64 (+0.59%)

At close: November 11 4:00 PM EST

Summary | Conversations | Statistics | Profile | Financials | Options | Holders | Historical Data | Analysts

November 18, 2016 | In The Money | Show: List | Straddle | Lookup Option

Calls for November 18, 2016

Strike	Contract Name	Last Price	Bid	Ask	Change	% Change	Volume	Open Interest	Implied Volatility
60.00	AAPL161118C00060000	51.95	48.55	49.30	0.00	0.00%	7	0	270.12%
65.00	AAPL161118C00065000	47.04	43.55	44.30	0.00	0.00%	113	0	238.28%
70.00	AAPL161118C00070000	38.51	38.15	38.65	-3.49	-8.31%	2	20	179.49%
75.00	AAPL161118C00075000	37.02	33.55	34.30	0.00	0.00%	19	8	180.86%
80.00	AAPL161118C00080000	28.53	28.10	28.85	0.00	0.00%	12	144	103.13%
85.00	AAPL161118C00085000	23.13	23.20	23.65	-3.27	-12.39%	5	13	108.98%
90.00	AAPL161118C00090000	18.49	18.35	18.65	-1.44	-7.23%	10	132	71.48%
92.50	AAPL161118C00092500	15.92	15.80	16.30	-1.67	-9.49%	1	101	68.16%
95.00	AAPL161118C00095000	13.36	13.40	13.65	-1.34	-9.12%	13	153	56.06%
97.50	AAPL161118C00097500	13.90	13.60	13.85	1.25	9.88%	5	109	131.89%
99.00	AAPL161118C00099000	9.80	9.45	9.70	0.00	0.00%	88	66	51.86%
100.00	AAPL161118C00100000	8.30	8.45	8.70	-2.60	-23.85%	111	1,837	47.46%
101.00	AAPL161118C00101000	7.23	7.50	7.75	-3.41	-32.05%	3	2	45.07%
102.00	AAPL161118C00102000	6.70	6.55	6.75	-2.54	-27.49%	30	200	40.43%
105.00	AAPL161118C00105000	3.60	3.80	3.95	-2.31	-39.09%	1,593	4,014	31.45%
106.00	AAPL161118C00106000	3.10	2.98	3.10	-2.00	-39.22%	289	592	29.18%
107.00	AAPL161118C00107000	2.29	2.24	2.29	-1.75	-43.32%	1,540	1,857	26.61%
108.00	AAPL161118C00108000	1.62	1.60	1.64	-1.83	-53.04%	16,086	4,132	25.56%

I mentioned earlier that Mathematica's FinancialData doesn't provide access to Yahoo Finance data for options. Neither does the Matlab Fetch function. User solutions exist to solve the problem, but unfortunately these tend to be poorly maintained and break down whenever Yahoo changes the specification for its api. Thus, for instance, the GetOptionChain utility provided in the Mathworks FileExchange forum (see https://uk.mathworks.com/matlabcentral/fileexchange/50455-getyahoooptionchain) produces an error in the form 'undefined function or variable "yyyymmdd"'. Likewise the solution offered in the *Making a Stock Options Database in Mathematica* post in Mathematica StackExchange (http://mathematica.stackexchange.com/questions/28053/making-a-stock-options-database-in-mathematica) is several years out of date and no longer operational. Both of these malfunctions result from recent changes in the Yahoo api. Of course, such problems are often trivial and easily solved - but at other times the challenge might be significant. If you decide to go down this route you must reconcile yourself to periods without available options data, or dedicating several hours to unraveling the most recent api changes and repairing the code.

Depending on how extensive your data requirements are, one solution might be to access options data via the api of your brokerage platform. In principle this will allow you to gather not only historical data but also real-time data, albeit at some latency. The prices retrieved by this means are often less stale than those obtained via Yahoo. But the downside is that the limitations of data access from brokerage platforms mean that you will be unable to obtain the option chain for more than a handful of securities, without paying substantial additional fees.

An alternative solution that recently caught my eye is Stock Data Solutions (https://www.stock-data-solutions.com/) . The developers have created an RTD server enabling the user to access both historical and real-time data from Yahoo, Google and MSN in MS-Excel, including both stock and options data. I have been checking out the service and, so far, with the exception of a couple of glitches, it appears to work as advertised.

Conclusion

For basic historical stock information both Mathematica and Matlab offer simple interfaces that enable you to download voluminous quantities of data for free from Yahoo or Google Finance, where it can be used in-memory and/or exported easily in a variety of formats suitable for database operations. If you require intraday updates a simple timer function will allow you to refresh the data at regular intervals and this may sufficient for your analytical purposes. For more latency sensitive trading applications it may be necessary to subscribe to realtime data services, such as Mathematica's finance platform, for example. However, some low-cost alternatives exist which may be sufficient for intraday trading strategies that are not high frequency.