ROOT IN FINANCE

A HEDGE FUND PERSPECTIVE

Jason Ward

ROOT Users Workshop, Saas Fee, 11-14 March 2013.

BACKGROUND

- 1992-1996 PhD on OPAL (University College London)
    - Photon structure function from 2-photon collisions
    - FORTRAN / PAW
- 1996-1999 Postdoc (University of Glasgow)
    - Predominantly ALEPH (W Mass): FORTRAN / PAW / Linux Cluster
    - ~8 months on ATLAS (SCT): First time using C++/ROOT
- 2000-2002 CERN Fellowship
    - Continued on ALEPH (W Mass): FORTRAN / PAW / Perl

★ Observations:

- "Large" collaborations - lots of support.
- Very oriented towards data analysis.
- Built a sense of what PAW/ROOT can do.

BACKGROUND

- One of the key skills for an aspiring Quantitative Analyst in finance was/is to program in C++.
    - Fast enough for some real-time financial applications. Especially useful if doing pricing in an Investment Bank. Large institutions with extensive C++ libraries.
    - Language of choice became C++. Therefore inclined to using ROOT.
- 2003-2013 Quantitative Researcher at a London-based Hedge Fund.
    - Core task was to devise systematic trading strategies.
        - Systematic ⇒ testing strategies on historical financial data. Data analysis!

SMALL BUSINESS

★ Vastly different from Big Science!

- May be the only ROOT user.
    - Support yourself.
    - Emphasis on (a) software interfacing and (b) communicating results.
- Resilient to business lifecycle. Change can be rapid.
- Advances in software/hardware technology. Platform changes.
- Might have confidentiality constraints.

FUTURES CONTRACTS

- These contracts are very liquid and transparent.
- Standardised contract to buy/sell a specified asset, of a standardised quantity, for a price agreed upon today with a delivery and payment at a specified future date.
- Exchange traded.
- If you are buying (selling) the underlying asset in the future the buyer (seller) of the contract is said to be "long" ("short").
- "Long" ("short") ⇒ hope for a price rise (fall).
- Example: E-mini S&P 500 Futures:
  - Traded on the CME Globex platform;
  - Contract Size = $50 × S&P 500 Index;
  - Trading can occur up to 8.30 a.m. (Central Time) on the 3rd Friday of the contract month;
  - Contract months: Five months in the March Quarterly Cycle (Mar, Jun, Sep, Dec).

SOME EXAMPLES OF FUTURES CONTRACTS

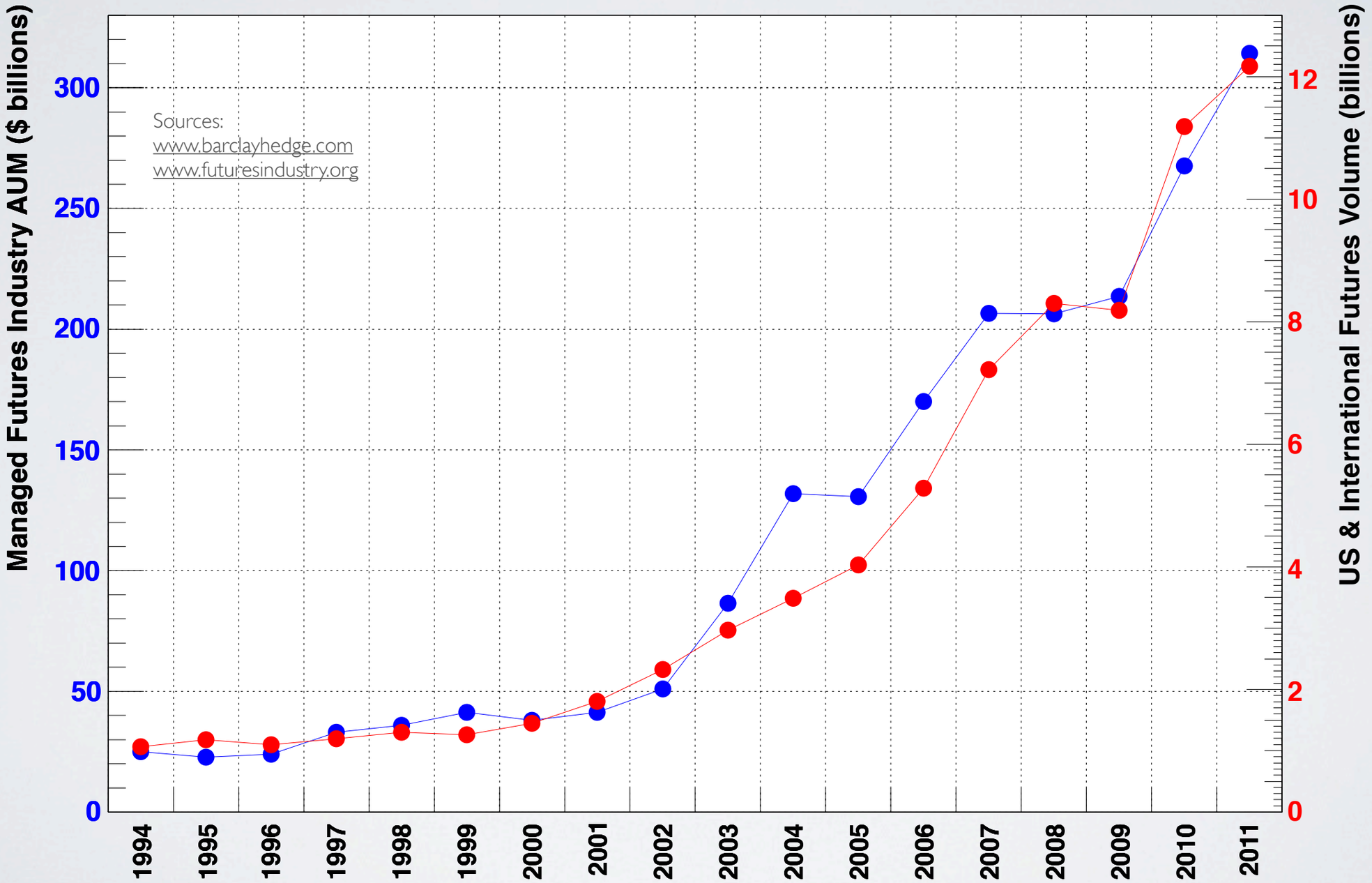
Indices	Bonds and Interest Rates	Commodities
CAC40	EuroDollar	WTI Crude
DAX	Euro - Bobl	Heating Oil
Eurostoxx	Euro - Bund	Natural Gas
FTSE	US 5 Year	Gold
Hang Seng	US 10 Year	Silver
NASDAQ	US Long Bond	Soybean
Nikkei 225	Long Gilt	Corn
S&P 500	90 Day Sterling	Wheat

- Just a sample of various markets that have futures contracts.
- Need to prepare to analyse portfolios (and sectors) of instruments.

GROWTH OF CTAs & VOLUME

CTA = Commodity Trading Advisor

Asset manager dealing in futures contracts and options on futures contracts.



PORTFOLIO BACKTESTING

★ Overall aim: Develop, build and test a robust trading algorithm before implementing it.

- Choose instruments to go into a portfolio.
- Imagine (for illustration) that you are working with daily data (i.e. closing price every day for each instrument).
- Imagine trying out an N-day Simple Moving Average (SMAV) strategy:
    - In each instrument:
        - Buy if Price today > SMAV[N](Price)
        - Sell if Price today < SMAV[N](Price)
        - Re-assess position one period (in this case, day) later
- Weight the instruments in the portfolio. Determine P&L each day for the portfolio.
- Do so in a statistically significant way, with nice risk characteristics.
- Avoid over-fitting the data.
- Monitor the performance.

DATA ANALYSIS

- Preparation
  - Choose | Purchase and/or Record
- Preprocessing
  - Reformat | Clean | Filter/Cuts | Align | Transform
- Analysis
  - Visualisation | Volatility & Correlation | Regression | Portfolio Statistics
- Post-processing
  - Documentation | Communication

MATLAB

- Now available: R2012b.
    - New MATLAB Desktop.
- Serves many aspects of the financial industry; Popular for backtesting.
- Webinars, Blogs, Newsgroups, File Exchange, Documentation Center etc.
    - The answers are always there. Good for an independent worker.
- Some features:
    - Trading rule selection via Genetic Programming.
    - Parallel computing.
    - Code analyzer report. Displays potential errors and problems and suggests improvements.
    - Profiler for improving performance.
- Many toolboxes. Purchase each one separately.
- [Octave is the open source equivalent].

S / R / S-PLUS

- Both R and S-PLUS are two modern implementations of the statistical programming language S.

- R is open source. S-PLUS is commercial.

- I tried S-PLUS for a few of years:

    - Prior to the S-PLUS 7.0 release in 2005 when they introduced functionality for working with large datasets.

    - If language is unfamiliar and you need something done, may have to hire/ contract in the expertise.

PYTHON AND PANDAS

- Python very popular, with strong growth over last ~10 years.
  - Low entry barrier. Fairly simple and powerful.
  - Looks nicer compared to Perl!
- Good integration.
- [Slower].
- See "Python for Data Analysis" by Wes McKinney (published by O'Reilly, 2013).
  - Main author of the pandas library
  - http://pandas.pydata.org
  - Background in fund management. Initial pandas design for financial data analysis.

SOME OTHERS

- Bloomberg Custom Technical Studies.
    - Code algorithms in C# / VBA to run on portfolios of instruments.
    - Suggests that backtesting portfolios is becoming more in demand.

APPEAL OF USING ROOT

★ Appreciate these points when you don't have them!

★ Onlookers impressed by points underlined.

- Good data visualisation; Fast for dense data.
    - Especially so in interactive mode when exploring/understanding data.
    - Spot check *anything* by hand.
- Open-source.
- Write own code: programming flexibility.
    - Custom strategies.
    - Fold in derived quantities.
- Various operating systems.
- Data compression, storage and portability.
- Directory structure and various object types stored in one root file.
    - Easy to find various kinds of information when you need it.
- Easy to apply Monte Carlo methods.

CHALLENGE OF USING ROOT

★ Time series analysis!

- Requires good date/time functionality.
- In HEP, the events are independent. In predictive finance, the key relationship being sought after is across time (predicting future price moves).
- Understanding contemporaneous relationships are important too, but require a lot of work to acquire and align financial data.

★ Integration in a non-ROOT environment.

STYLE OF USING ROOT

- End user!
- Compiled into an executable:
  - Minimal use of CINT.
  - Warning and error messages from compilation very helpful.
- Used standard C++ and STL where possible:
  - May have to use pieces of code elsewhere without ROOT.
  - Clearer to people who don't use ROOT.
- Got data into a TTree, then wrote functions that operated on the TTree:
  - Operation on time series common and re-usable.

GETTING DATA INTO TTREE

- Big positive payoff to getting data into TTrees a.s.a.p.
    - However, getting data prepared for ReadFile can be an effort.
- As many data formats as there are vendors/sources.
- Common delivery of data:
    - Flat files | Excel spreadsheets.
- Varying degrees of readability, cleanliness and Date/Time formats.
- Need to deal with things like:
    - N/A | Chars where numbers should be | Missing fields | ...
    - Strings and strings with spaces are very common.
- ⇒ Python!

★ In ReadFile method:

- ★ "#" is very handy!
- ★ Thank you for .csv update.
- ★ Any development/examples dealing with getting data into TTrees very welcome!

MISSING / INCALCULABLE DATA

- Imagine e.g. Date/I:Price/D
  - Missing days (e.g. holiday days)
    - Omit | backfill | ...
  - Dirty data (just one bad point can be destructive, e.g. decimal point in wrong place)
    - Exclude
    - Estimate replacement. e.g. AUD/NZD = (AUD/USD) × (USD/NZD)
- Imagine e.g. Date/I:Open/D:High/D:Low/D:Close/D:Volume/I:OpenInterest/I
  - Subsection of entry may be missing e.g. "Date,Open,High,Low,Close,,"
  - Internal consistency checks e.g. Close ≤ High.
- NaN (in both primary data and derived quantities e.g. moving average)
  - Very one-dimensional
  - Can unwittingly propagate across and inside events
- One sledgehammer solution:
  - Pair every data item to a status value. Could be heavy for high frequency data, but for lower frequency data one has full control and ability to understand how data issues propagate to the final results.

DATES AND TIMES

- Crucial to understand date and time stamps:
    - Data come from various time zones. Great care required to ensure no look-ahead in time in cross-relational studies.
    - Requires a perfect understanding of UTC offsets historically!
- TDatime is *not* so useful:
    - Constrained to using dates ≥19950101.
    - No support for time zones.
- Only remaining option in ROOT is TTimeStamp.
    - Write extra functionality yourself;
    - Integrate with other functionality (Python, Quantlib...)
        - [Quantlib is an open-source library for Quantitative Finance: requires Boost]
- A lot of functionality depends on basic date and time manipulation.
    - e.g. aggregating daily data into weekly or monthly data;
    - TGraph with x-axis as date/time is one of the most important graphs.

TTimeStamp

- Seems to miss methods that exist in TDatime like GetDay(), GetMonth(), GetSecond(), GetMinute(), GetHour() etc.

- Quantlib::Date has useful methods like:

  - Last day of the month to which the given date belongs (endOfMonth).

  - Whether a date is the last day of its month (isEndOfMonth).

  - Next given weekday following or equal to the given date (nextWeekday).

  - n-th given weekday in the given month and year (nthWeekday).

- Example from Quantlib::Calendar:

  - Number of business days between two given dates (businessDaysBetween).

- Date as a single integer (yyyymmdd) and time as a single integer (hhmmss) very useful in a TTree as a double index.

- Conversion of TTimeStamp into many different string formats.

- Date/time functionality from (TDatime ∪ Quantlib ∪ Python) might be a good guideline, with ability to use in TGraph.

COMMENTS / THOUGHTS (1)

- Web visualisation! Will have a huge impact...
  - Share contents of a root file without recipient needing to install ROOT.
  - Ability of user to interact with graphs e.g. zoom in on a TGraph.
  - ToolTip information useful to identify certain points in a time series.
  - LaTeX.
  - Design for the end user.

COMMENTS / THOUGHTS (2)

- Time series ⇒ many derived quantities.
  - Daily returns
  - Moving window calculations (e.g. SMAV)
  - Many many other things...
- Adding branches to a TTree is useful.
  - Any advances welcome! Especially anything that makes the procedure foolproof.
- Friendship between TTrees that are not aligned?

COMMENTS / THOUGHTS (3)

- TGraph (and TMultiGraph) are important as they are the graphical representations of time series. Any improvements to interactivity are welcome.
    - Joint (x, y) zoom.
    - Ability to shuffle around left/right/up/down.
    - When hovering over a point, show x-value in a date/time format.

COMMENTS / THOUGHTS (4)

- Cleaning very dense time-series data.
  - Can develop algorithms to suggest possible dirty data.
  - Cannot fully replace human judgement.
- Emphasises need for graphical interaction with data to mark up various problems.
  - Interesting to hear from people who have to view/clean large datasets near the front end.
  - Editable TTree?

COMMENTS / THOUGHTS (5)

- The more examples, the better!
  - Just seeing a “trick”/solution/application is all that is needed.
  - Thank you (Eddy!) for portfolio.C in tutorials/quadp
- More workshops!
- Development of Python related tools very important.

SUMMARY

- ROOT is a very effective tool for financial data analysis, especially for high density data, but it requires an investment from the user(s) to do some coding work and become familiar with what is available.

- This workshop is very exciting because many advances are being made.

- Many tools are available for financial data analysis. Optimal choice depends on the business, the tasks at hand and the user's preferences.

- Thank you!