Building Market Data Interfaces For R

Rory Winston

rory@theresearchkitchen.com

Contents

About Me

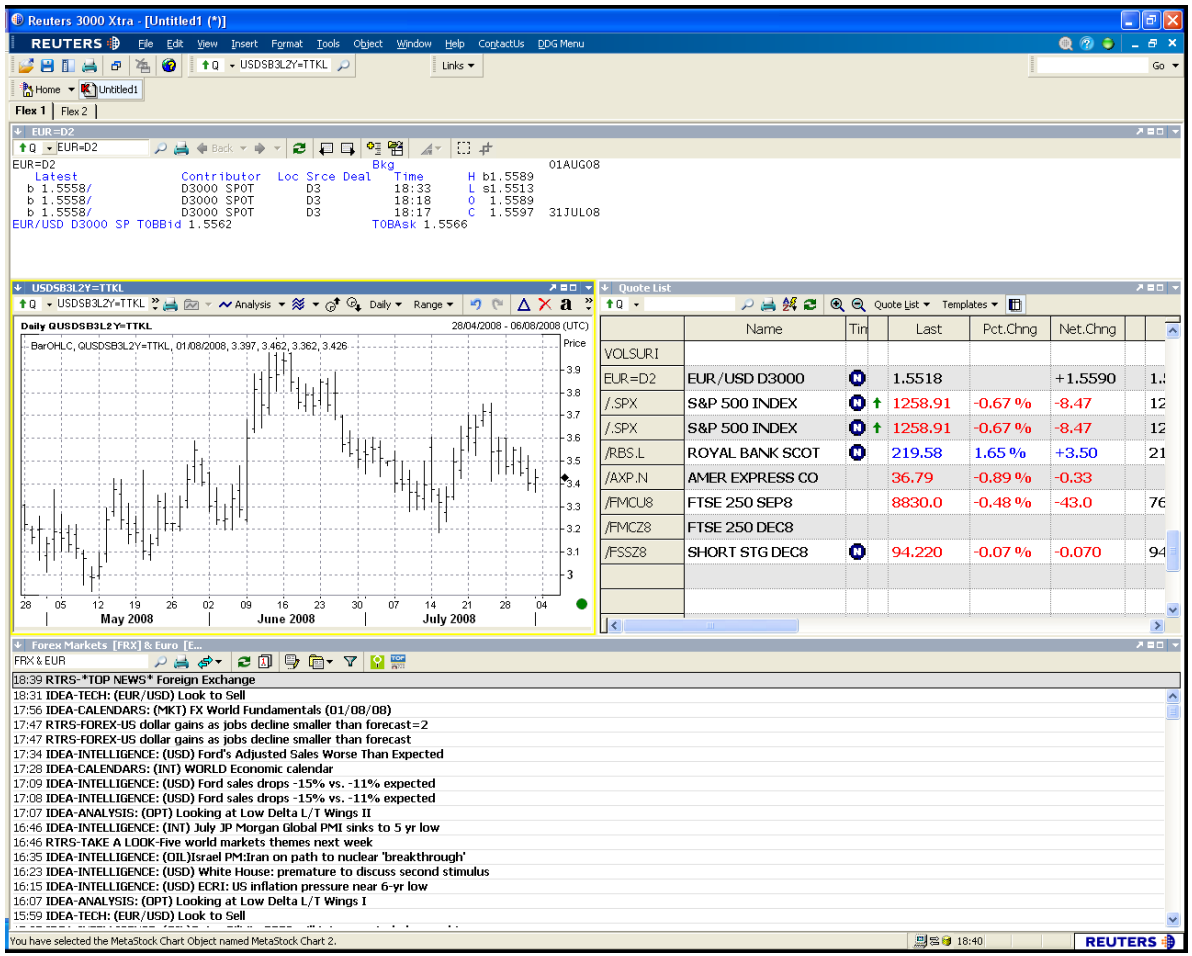
- Independent Software Consultant
- M.Sc. Applied Computing, 2000
- M.Sc. Finance, 2008
- Working in the financial sector, building trading systems in Java/C++
- Interested in practical applications of functional languages and machine learning
- Relatively recent convert to R

The Reuters Marketfeed System

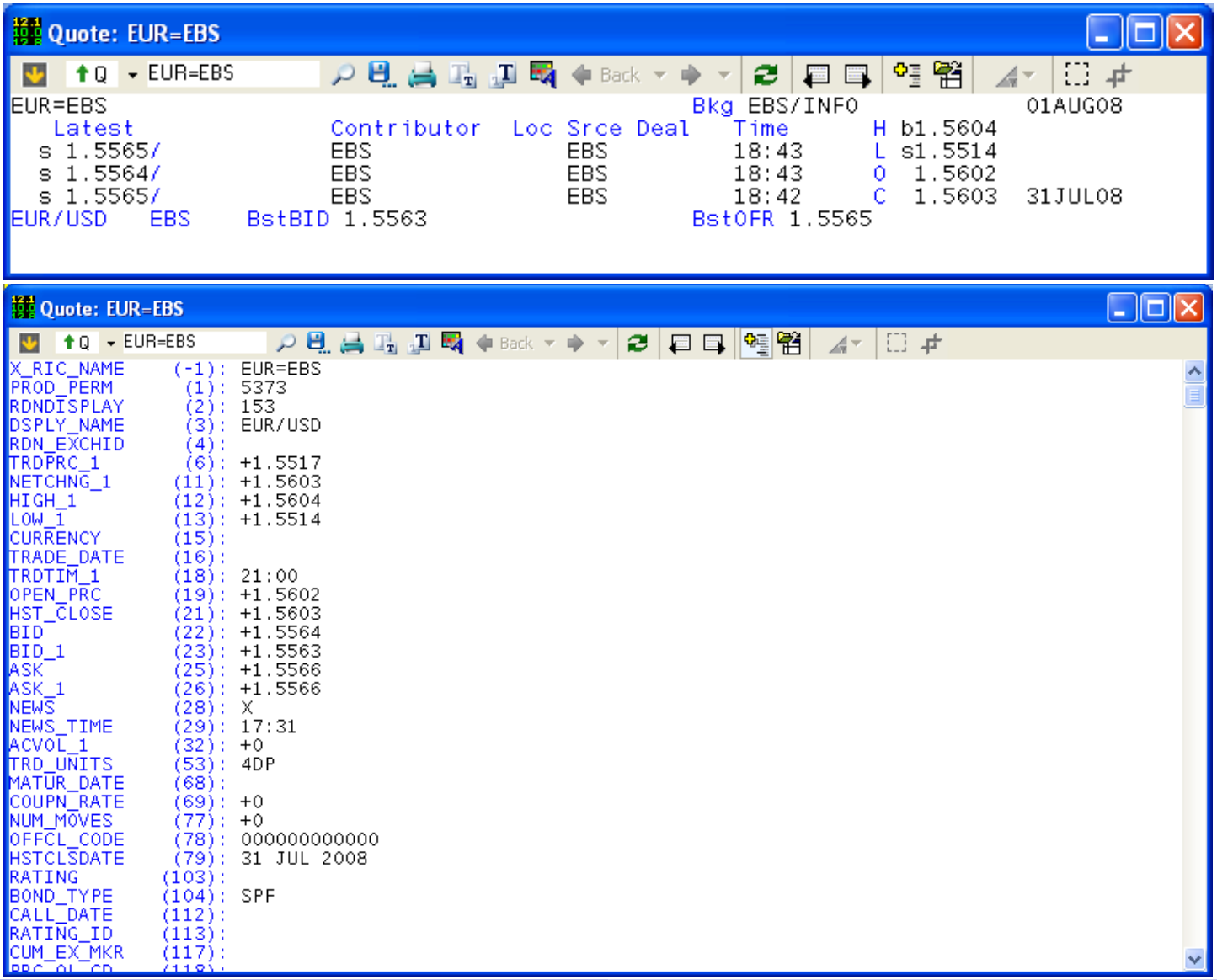
- Ubiquitous (especially in Foreign Exchange)
- APIs, Excel plugins, graphical workbenches, and trading GUIs
- Used by traders, structurers, quants, analysts....
- A market data item is identified by a RIC (Reuters Information Code)
- Format: `CODE=[specifiers]`
- Examples (FX): `EUR=`, `EUR=EBS`, `GBPJPY=D2`
- Examples (Equities): `MSFT.O`, `IBM.N`
- Examples (Fixed Income): `EUR3M=`, `US30YT=RR`
- Each market data item contains a number of data fields, e.g. `BID`, `ASK`

Reuters Desktop

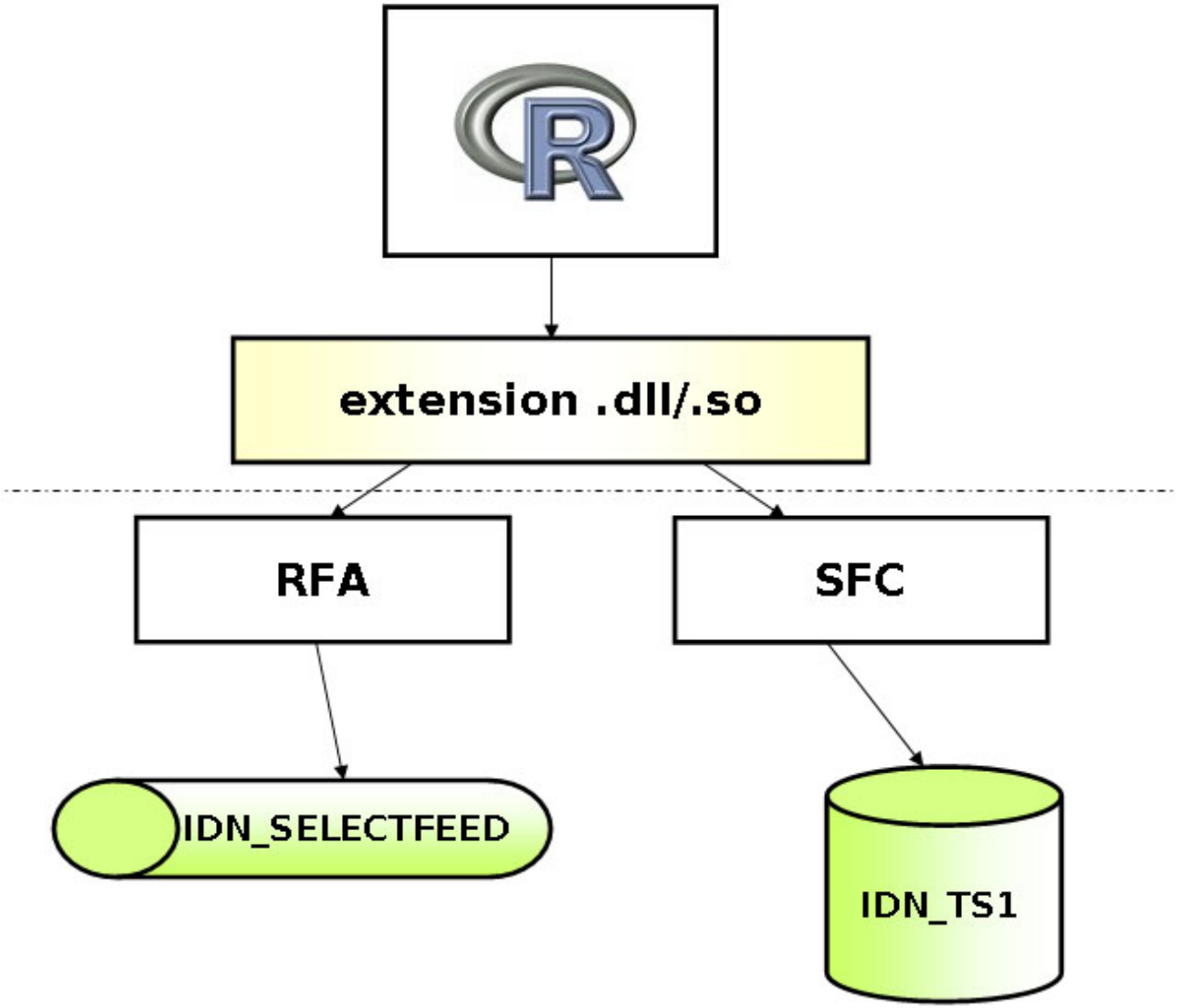
Example Reuters 3000 session:



Reuters Interfaces



Extension Architecture



Potential Uses:

- Testing real-time trading algorithms
- Market data collection and validation
- Dynamic portfolio optimization algorithms
- Trading cost / spread analysis
- Real-time interest rate curve building
- Econometric analysis
- Applying R's vast library of computational statistics and machine learning routines to tick data
- Turn R into a "trading workbench"

Example - Real Time Subscription

Real-time interface:

```
rsub <- function(duration, items, callback)
```

The call `rsub` will subscribe to the specified rate(s) for the duration of time specified by `duration` (ms). When a tick arrives, the callback function `callback` is invoked, with a data frame containing the fields specified in `items`.

Multiple market data items may be subscribed to, and any combination of fields may be be specified.

Uses the underlying RFA API, which provides a C++ interface to real-time market updates.

Real-Time Example

```
# Specify field names to retrieve
fields <- c("BID","ASK","TIMCOR")

# Subscribe to EUR/USD and GBP/USD ticks
items <- list()
items[[1]] <- c("IDN_SELECTFEED", "EUR=", fields)
items[[2]] <- c("IDN_SELECTFEED", "GBP=", fields)

# Simple Callback Function
f <- function(df) { print(paste("Received",df)) }

# Subscribe for 1 hour
ONE_HOUR <- 1000*(60)^2
rsub(ONE_HOUR, items, callback)
```

Real-Time C++ Interface

I used the `.Call` interface, as it enabled me to easily deal with the data frame arguments and callback functions.

```
DL_EXPORT SEXP subscribe(SEXP t, SEXP spec, SEXP callback) {
...
}
```

Boost provides nice type-based visitor functionality, which I use to convert between Reuters wire-level data type and native R SEXP-based types.

Example - Historical Fetch

Historical interface:

```
fetch <- function(code, n=0, t="d", sd=NULL, ed=NULL)
```

Returns a data frame with data specified by `code`. The number of items returned can be specified by `n`, or it is defined as the number of items of the specified periodicity `t` (e.g. `"d"` = "daily") between start date `sd` and end date `ed`.

Uses the Reuters SFC API to obtain historical data.

Historical Interface Examples

```
> fetch("IBM.N", n=20)
        Date    CLS    OPN     HI     LO     VOL     VWAP
1  2008-07-31 127.98 127.77 129.50 127.76 2334800 128.4172
2  2008-07-30 128.86 128.42 129.00 127.10 1936400 128.2872
...
> fetch("EUR=", sd="1/1/2007", ed="1/1/2008")
         Date    BID    OPN     HI     LO    ASK
1   2008-01-01 1.4587 1.4585 1.4608 1.4576 1.4591
2   2007-12-31 1.4589 1.4719 1.4747 1.4568 1.4592
...
> fetch("EUR3M=", n=20, t="w")
        Date    BID    OPN     HI     LO    ASK
1  2008-07-25 -75.60 -77.60 -74.96 -79.30 -74.41
2  2008-07-18 -76.94 -76.30 -75.44 -81.10 -76.04
...
```

Historical Interface Example

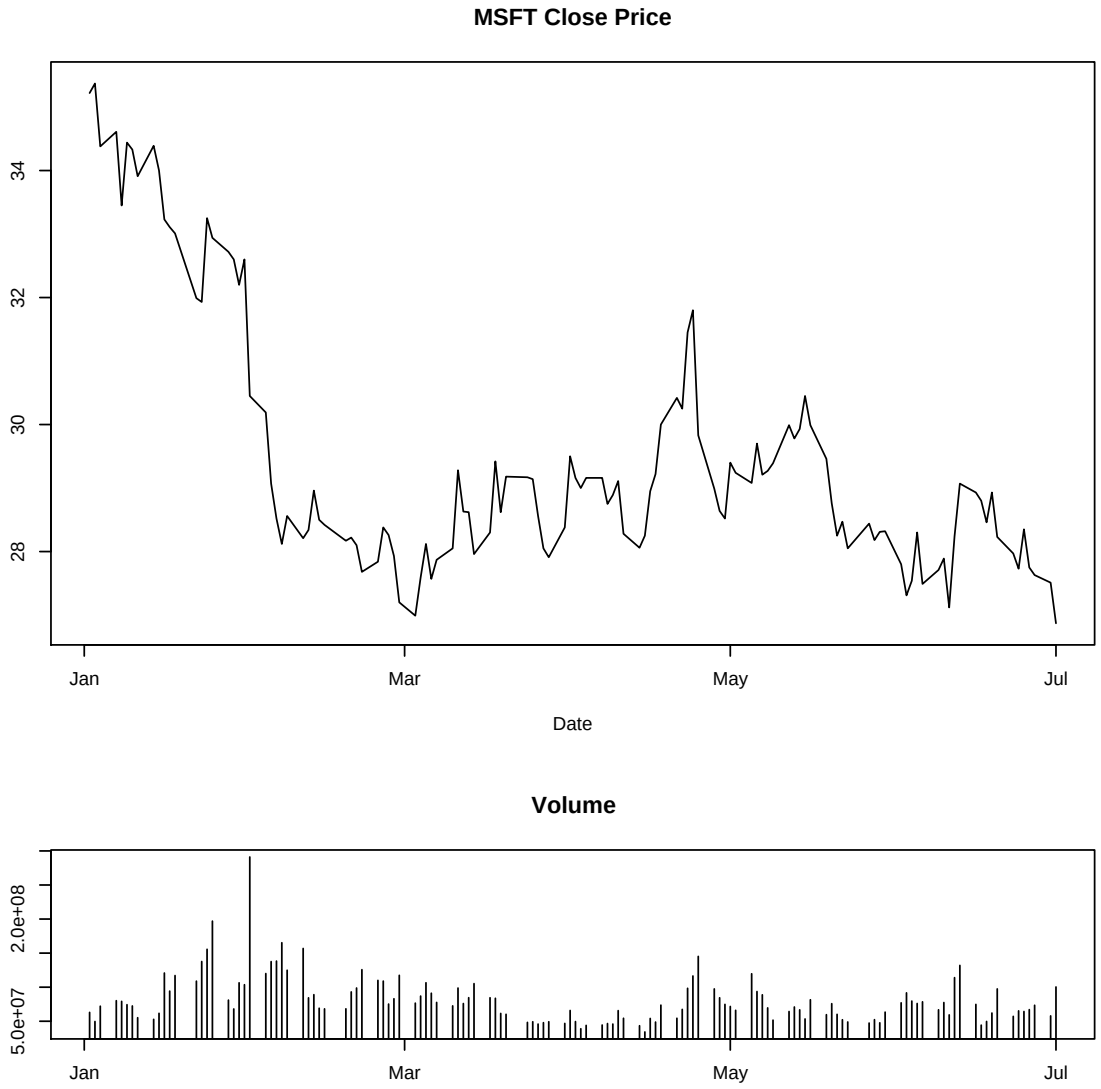
```
msft.vol <- fetch("MSFT.O",sd="1/1/2008",ed="7/1/2008")

layout(matrix(c(1,1,1,1,2,2), 3, 2, T))

plot(msft.vol$Date, msft.vol$CLS,
 main="MSFT Close Price", type="l", ylab="", xlab="Date")

plot(msft.vol$Date, msft.vol$VOL,
 main="Volume", type='h', ylab="", xlab="")
```

Historical Interface Example



Historical Fetch C++ Interface

Again using the `.Call` interface:

```
DL_EXPORT SEXP fetch(SEXP item, SEXP nItems,
 SEXP intervalCode, SEXP startDate, SEXP endDate) {
...
```

Code handles missing dates (e.g. market holidays), and coercion of underlying datatypes to appropriate R datatypes

Notes on Implementation

- Reuters Issues:
  - Reuters APIs are generally awful (newer APIs, i.e. RFA are slightly better)
  - Confusing / inconsistent functionality; Peculiarities with C++ runtime
- R-related issues:
  - Generally very few, once you get used to the idiosyncratic "Lisp-via-C-macros" style;
  - Some issues with asynchronous real-time updates and the single-threaded R interpreter
- The combination of Boost, C++ and R is extremely powerful
- Combining this with market data functionality adds a new dimension of capabilities
- FX, options, futures, forwards, interest rates, bonds, indices, equities...even news updates

Future Work

There are still some issues to work on, mainly some configuration and threading issues in the real-time extension

Mathworks bundle a number of financial data feed providers in their DataFeed toolbox for Matlab:
`http://www.mathworks.com/products/datafeed/`
Maybe a similar extension (bundling say, connections to Reuters, Bloomberg, Lim, Thomson, etc) for R may be a useful addition?

A financial "showcase application" for R would be great: maybe a financial analysts' toolkit, similar to the Holt system (combining elements of statistical learning and quantitative accounting analysis).