Programming with financial data: Connecting R to Lim and Bloomberg

Dirk Eddelbuettel
B of A and Debian
edd@debian.org

Presentation at the
conference, Vienna, May 20-22, 2004
In finance, the status quo is a spreadsheet application – a framework many users are familiar with.

Various add-ons enhance spreadsheets with specialised functionality, including access to data repositories or live feeds.

While we feel that data can be analysed more effectively in a data programming environment such as \texttt{R}, we must be able to provide comparable direct data access methods.

Here we provide an overview of two (local and as yet unreleased) packages that allow access to data coming from MIM, the Market Information Machine by LIM (a specialised data base vendor), or Bloomberg Terminal, an integrated application provided by Bloomberg.
Lim: Overview

- Specialised database (and content) provider:
  - Focus on time-series storage and retrieval.
  - Hierarchical, not relational.
  - Fast for retrieval and queries.
  - Popular with financial and energy firms.
  - Backend servers on Solaris or Linux.
- At B of A used for approximately 650,000 series.
Lim: Existing Interfaces

Lim provides several front-end programs (standalone Motif/X11 or Java; Excel plugin), and several programming language interfaces, (C, Java, Perl, VB, ..., as well as Matlab and S-Plus).

Based on the fairly simple S-Plus interface, it was reasonably straightforward to derive an R interface ...

... given that the C interface for R is documented, and illustrated with numerous examples in other CRAN packages.

Our implementation currently provides efficient access (at the C API level) to four basic sets of LIM functions.
**Lim: getLimData**

getLimData retrieves a contiguous block of data for one or several symbols.

```r
> args(getLimData)
function (relations = c("FV", "TY"), columns = c("Open", "High", "Low", "Close"), from.date = ISOdate(2002, 1, 1), to.date = Sys.time(), host = "chissd27", port = 0, exec.units = 4, num.exec.units = 1, rtd.use = 0, holiday.fill = 0, missing.fill = 0, its = FALSE)
NULL
> Data <- getLimData(c("FV"), from.date = ISOdate(2000, 1, 1), its=TRUE)
> Data[1:3,]
     Open     High     Low     Close
2000-01-03 97.78125 97.81250 97.51563 97.59375
2000-01-04 97.76563 97.96875 97.65625 97.92188
2000-01-05 97.78125 97.93750 97.48438 97.54688
> dim(Data)
[1] 1089     4
```
**Lim: getLimQuery**

`getLimQuery` executes a query in the Lim Query language which allows fairly general statements of the ’show X when Y’ type for various simple or complex conditions. As with `getLimData`, various time resolutions can be specified, and daily is the default.

```r
> args(getLimQuery)
function (query, host = "chissd27", port = 0, exec.units = 4,
    num.exec.units = 1, rtd.use = 0, holiday.fill = 0,
    missing.fill = 0, its = FALSE)
NULL
> print(getLimQuery("Show Fives: FV Tens: TY \n
    when date is within 1 week"))

   Fives   Tens
2004-04-30 109.9375 110.5000
2004-05-03 109.9063 110.5000
2004-05-04 109.8281 110.2969
2004-05-05 109.7031 110.0625
2004-05-06 109.5313 109.8281
```

Programming with financial data: Connecting R to Lim and Bloomberg – p. 6
**Lim: getLimRelationInfo**

getLimRelationInfo provides information about the possible ’child’ relations in the hierarchical ’tree’ of Lim symbols; this is e.g. used to traverse the ’tree’ of all available symbols in a simple tcl/tk browser using a tree widget.

```r
> args(getLimRelationInfo)
function (relation = "VIX", host = "chissd27", port = 0) NULL
> getLimRelationInfo()
$status
[1] 2

$children
character(0)

$relationpath
[1] "TopRelation:Indices:IndexOptions:CBOE_Options"

$parentrelation
[1] "CBOE_Options"
```
**Lim: getLimInfo**

`getLimInfo` provides information about data availability (both intra-day and daily) for a given symbol; it returns data frames with start and end dates for every available column.

```r
> args(getLimInfo)
function (relation = "BOA.VIX", host = "chisbd27", port = 0) NULL
> getLimInfo("TY_2004M")
$TY_2004M
daily

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>2003-10-02 2004-05-06</td>
</tr>
<tr>
<td>High</td>
<td>2003-10-02 2004-05-06</td>
</tr>
<tr>
<td>Low</td>
<td>2003-10-02 2004-05-06</td>
</tr>
<tr>
<td>Close</td>
<td>2003-10-02 2004-05-06</td>
</tr>
<tr>
<td>Volume</td>
<td>2003-10-02 2004-05-05</td>
</tr>
<tr>
<td>OpenInterest</td>
<td>2003-10-02 2004-05-05</td>
</tr>
<tr>
<td>HistVol</td>
<td>2003-11-17 2004-05-06</td>
</tr>
<tr>
<td>ImpVol</td>
<td>2003-11-17 2004-05-06</td>
</tr>
</tbody>
</table>
```

[...]
**Lim: Performance**

Performance is generally fairly good. The backends perform load-balancing and distributes the request from a master server to the ‘best’ slave server. However, network latency can be a factor as in the fifth attempt below.

```r
> for (i in 1:10) cat(system.time(X<-getLimData("TY",
   from.date=ISOdate(1994,5,1))),"\n")
0.05 0.01 0.15 NA NA
0.03 0.01 0.11 NA NA
0.03 0.01 0.12 NA NA
0.04 0.02 0.13 NA NA
0.34 0.06 3.15 NA NA
0.02 0.01 0.12 NA NA
0.02 0.02 0.11 NA NA
0.06 0.01 0.13 NA NA
0.03 0.01 0.12 NA NA
0.02 0.01 0.12 NA NA
> dim(X)
[1] 2510  4
```
On Windows, **R** requires the MinGW variant of the GNU C compiler.

In order to build with a Microsoft Visual C-compiled object code library, the `reimp` tool can transform the supplied library into one suitable for building with MinGW, see [http://www.mingw.org/mingwfaq.shtml#faq-msvcdll](http://www.mingw.org/mingwfaq.shtml#faq-msvcdll).

On Unix/Solaris, we were able to use source for the LIM API interface, so building is just like building from any other collection of source files.
Specialised content provider.

Very popular and trusted in the financial industry, also very expensive (approx. $1500/month).

Truly vast scope of data and analytical functionality, delivered through *Bloomberg Terminals* which (these days) is a Windows application program.

As illustration of scope, consider that there are almost 11,000 fields in the file `Bbfields.tbl` which define mnemonic codes for the different ‘columns’ of data that can be downloaded:

10|Real Time Quotes|0|0001|Last Price|LAST_PRICE|3998|27|3
10|Real Time Quotes|0|0002|Bid Price|BID|3998|27|3
[...]
128|Ratings|0|0590|S&P Rating|RTG_SP|2520|4|1
[...]
185|Descriptive Info|0|0E68|Instit Owner (# of Buyers)|EQY_INST_BUYS|4|4
[...]
Bloomberg provides an Excel add-in as well ActiveX bindings for Visual Basic, Microsoft MFC, PowerBuilder, Delphi, J++, Access VBA and Javascript.

Less well known is a plain C API which we prefer to use.

C API documentation comes with fifteen complete examples and fairly detailed documentation.

See \texttt{WAPI <GO>} and follow links for ’Bloomberg C API’; downloads do not seem to be available from outside the Bloomberg terminal.

Example programs operate in asynchronous mode: connection requests are formed, sent and then polled for data — this needs to broken up for coding suitable for \texttt{R} add-ins.
The `getBloombergStatic` function can be used to retrieve a set of Bloomberg 'fields' for a set of Bloomberg 'tickers'.

```r
> args(getBloombergStatic)
function (Securities = c("BAC Equity", "GT10 Govt", "EDA Comdty"),
    Fields = c("0x0452", "0x0454", "0x0553", "0x0555", "0x0556",
               "0x0557", "0x0558", "0x0560", "0x0574", ), verbose = FALSE)
NULL
> getBloombergStatic("GT10 Govt")

<table>
<thead>
<tr>
<th>NAME</th>
<th>TICKER</th>
<th>PX_BID</th>
<th>PX_ASK</th>
<th>PX_OPEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT10 Govt</td>
<td>US TREASURY N/B</td>
<td>T 93.85937500</td>
<td>93.89062500</td>
<td>93.87500000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PX_ASK</td>
<td>PX_OPEN</td>
<td>PX_HIGH</td>
</tr>
<tr>
<td>GT10 Govt</td>
<td>93.89062500</td>
<td>93.87500000</td>
<td>93.93750000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PX_LOW</td>
<td>PX_LAST</td>
<td>LAST_UPDATE</td>
</tr>
<tr>
<td>GT10 Govt</td>
<td>93.87500000</td>
<td>93.89062500</td>
<td>19:59:00</td>
<td></td>
</tr>
</tbody>
</table>
```

Programming with financial data: Connecting R to Lim and Bloomberg – p. 13
The `getBloombergHistory` function returns its objects (or a list of these in case of several input tickers) with columns corresponding to the supplied (or default) Bloomberg fields.

The is the workhorse function for daily data.

```r
> args(getBloombergHistory)
function (Securities = c("EDA Comdty", "BAC Equity", "GT10 Govt"),
         Fields = c("0x0001", "0x0553", "0x0555", "0x0559"),
         Currency = "USD", FromDate = Sys.time() - 24 * 60 * 60 * 5,
         ToDate = Sys.time(), verbose = FALSE, drop = TRUE)

NULL
> getBloombergHistory("EUR Curncy")

<table>
<thead>
<tr>
<th>LAST_PRICE</th>
<th>PX_BID</th>
<th>PX_ASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004-05-05</td>
<td>1.2175</td>
<td>1.2174</td>
</tr>
<tr>
<td>2004-05-06</td>
<td>1.2081</td>
<td>1.2081</td>
</tr>
<tr>
<td>2004-05-07</td>
<td>1.1883</td>
<td>1.1883</td>
</tr>
<tr>
<td>2004-05-10</td>
<td>1.1848</td>
<td>1.1849</td>
</tr>
</tbody>
</table>
```
The `getBloombergBarHistory` function returns intraday data for the given (one or several) Bloomberg ticker. Up to fifty calendar days can be retrieved.

```r
> args(getBloombergBarHistory)
function (Securities = c("EDA Comdty", "GT10 Govt", "JPY Curncy"),
         FromDate = Sys.time() - 48 * 60 * 60, ToDate = Sys.time(),
         barsize = 60, field = 1, verbose = FALSE, drop = TRUE)
NULL
> getBloombergBarHistory("EUR Curncy")
       Open  High  Low  Close Count  Volume Value
2004-05-09 21:03 1.18685 1.1875 1.1849 1.1867  1543    0     0
2004-05-09 22:03 1.1867 1.1867 1.1839 1.18435  1915    0     0
2004-05-09 23:03 1.18435 1.1845 1.18325 1.1838  1375    0     0
2004-05-10 00:03 1.1838 1.1847 1.1836 1.18375  1358    0     0
2004-05-10 01:03 1.1838 1.18485 1.1831 1.1847  2176    0     0
2004-05-10 02:03 1.18475 1.1859 1.18305 1.1837  3634    0     0
[...]
```
Bloomberg: Tick monitoring

- We also have a group of functions to initiate monitoring of real-time ticks.

- Via `initBloombergTickMonitor(Securities)`, monitoring is initiated for the given security symbols.

- Then, \( Z \leftarrow \) `getBloombergTickMonitor(\text{verbose=FALSE})` stores the current reading into a variable \( Z \).

- Finally, `closeBloombergTickMonitor()` ends the monitoring.

- This code is still somewhat untested.
Performance is fairly satisfactory, especially given that requests are handled remotely from wherever Bloomberg decides to answer them:

```r
> for (i in 1:10) cat(system.time(X<-getBloombergHistory("JPY Curncy",
                                           FromDate=ISOdate(1994,5,1))),"\n")
 0.39 0.04 1.71 NA NA
 0.44 0 1.29 NA NA
 0.69 0.03 1.57 NA NA
 0.39 0 1.22 NA NA
 0.41 0 1.54 NA NA
 0.71 0.02 1.55 NA NA
 0.41 0.02 1.75 NA NA
 0.42 0 1.26 NA NA
 0.73 0.01 1.61 NA NA
 0.39 0.01 1.29 NA NA
> dim(X)
[1] 2616   3
>
Building and running requires a valid Bloomberg data license, as well as the header and library files from the C API development kit.

The gcc compiler, in the MinGW variant that is used for building R, can link directly against the supplied bbapi.lib, and also needs to link with -lwsock32 to provide basic networking functionality.
Summary

As *Programming with Data* is such a powerful concept, in particular when it is implemented well as in the case of **R**, we like to apply it to the vast world of financial market data.

For *Programming with financial Data*, direct access methods for data repositories complete the data programming environment.

Based on a good year of usage, we have refined the Lim interface to **R**; coupling it with the *its* classes provided a very natural fit and made the use even easier.

Similarly, our experience with the Bloomberg interface to **R** has been good, but the code is still somewhat younger and will probably undergo more changes and enhancements.