

Package ‘RBloomberg’

April 17, 2009

Version 0.1-11

Date 2009-02-25

Title R/Bloomberg

Author Robert Sams <robert@sanctumfi.com>

Maintainer Robert Sams <robert@sanctumfi.com>

Description Fetch data from Bloomberg

License GPL

Depends RDCOMClient, zoo, chron

OS_type windows

Repository CRAN

Date/Publication 2009-02-27 13:20:10

R topics documented:

blpConnect	2
blpGetData	3
blpReadFields	5
lowlevel	6
Index	9

`blpConnect`*Open or shut a connection to Bloomberg*

Description

The function `blpConnect` returns a connection object to one of the Bloomberg interfaces. Currently, only the COM Desktop API is supported.

Usage

```
blpConnect(iface="COM", timeout=12000, show.days="week",
na.action="na", periodicity="daily")
blpDisconnect(conn)
```

Arguments

<code>iface</code>	Which Bloomberg interface? (only "COM" currently supported).
<code>timeout</code>	Timeout in Seconds
<code>show.days</code>	"trading", "week", or "all"
<code>na.action</code>	"bloomberg.handles", "previous.days", or "na"
<code>periodicity</code>	"daily", "weekly", "monthly", or "annual"
<code>conn</code>	Connection object

Details

Connection to other API's provided by Bloomberg (e.g., C Desktop API and C Server API) will be accessible via this function in future releases of this package. Currently, the object returned is of class 'BlpCOMConnect'.

```
## FIX ME ##
```

Author(s)

Robert Sams (robert@sanctumfi.com)

Examples

```
## by default establish connection via Desktop COM API
## Not run:
conn <- blpConnect()
blpDisconnect(conn)
## End(Not run)
```

blpGetData

*Get Bloomberg Data***Description**

This is the primary user-level function for retrieving Bloomberg data. Requests fall into four types: snapshot, historical, intraday bars, and intraday tick. The first type of call returns either static data or realtime data read at the time the function call is made. The remaining three call types return timeseries.

Usage

```
blpGetData(x, ...)
## Default S3 method:
blpGetData(x, ...)
## S3 method for class 'BlpCOMConnect':
blpGetData(x, securities, fields, start = NULL, end = NULL,
           barsize = NULL, barfields = NULL, retval = NULL, ...)
```

Arguments

<code>x</code>	connection object.
<code>securities</code>	vector of Bloomberg tickers.
<code>fields</code>	vector of field mnumonics. For intraday requests, this must be one or more of "BID", "ASK", and "LAST_PRICE".
<code>start,end</code>	chron object defining data window.
<code>barsize</code>	integer representing the duration (in clock minutes) of the bar intervals in an intraday bar timeseries request. If set to 0, the request is for intraday tick resolution. If NULL, the request is historical.
<code>barfields</code>	field aspect for intraday bar requests. One or more of "OPEN", "HIGH", "LOW", "LAST_PRICE", "NUMBER_TICKS", and "VOLUME". Number of fields returned is the length of <code>fields</code> times <code>barfields</code> .
<code>retval</code>	"matrix", "data.frame", "zoo", or "raw". If NULL, return a zoo for timeseries requests and a data.frame for snapshot requests; "raw" will return the uncoerced, nested list.
<code>...</code>	other args.

Details

The user specifies type of request through his choice of parameters. If no argument is passed to `start`, then a snapshot call is made. All three timeseries require that a chron object is passed to `start`. If nothing is passed to `barsize`, then a historical call is made. If `barsize=0`, a tick call is made. If `barsize` is passed an integer of 1 or greater, a bar call is made. All and only bar calls pass at least one mnumonic to `barfields`.

Conceptually, a tick is a timestamped (to the clock second) record of six fields containing bid/ask/last price and size. An intraday tick call returns an ordered timeseries of tick records. A tick exists when and only when at least one of the six fields changes values. Thus, for two consecutive ticks, T1 and T2, the values of the six fields in T1 can be assumed to persist throughout the period of time that elapses between the timestamps of T1 and T2.

An intraday bar call summarises (according to the mnemonic passed to `barfields`) the ticks that occur within fixed time intervals. The interval size is defined by `barsize`, which works at the clock minute resolution specified by the integer value passed to it.

Most of the information here is conjectural, based on trial-and-error usage of the Bloomberg API as opposed to references to the API documentation. You have been warned.

Bugs

There are no bugs of which I am aware. There are, however, a couple of idiosyncracies with the Bloomberg interface that you should keep in mind.

First, when `fields` contains "BID" and/or "ASK" and `barfields` contains "VOLUME", Bloomberg will return the sum of all bid/ask sizes, which is misleading. For example, if `barsize = 1`, the number of ticks for bar 18:30 is 3 and the ask sizes at each tick are 1000, 1020, and 1020, then the volume will be 3040. Use "VOLUME" combined with "NUMBER_TICKS" fields to back out an average bid/ask size.

Second, the "VOLUME" barfield returns a value of 0.00 instead of na's for bars containing no ticks. This is a problem for the special case whereby a bid or ask price is missing. For example, suppose you make a call with `fields="BID"` and `barfields="VOLUME"` and there is a bar in the return where `BID.VOLUME` is 0.00. That result is consistent with two scenarios: (1) the bid size and price in this bar is the same as it was in the previous bar and (2) there is *no* bid in this bar. To my knowledge, there is no way of determining which case holds.

Author(s)

Robert Sams (robert@sanctumfi.com)

Examples

```
## Not run:
conn <- blpConnect()

## Snapshot
eda <- blpGetData(conn, c("ED5 Comdty", "ED6 Comdty", "ED7 Comdty",
"ED8 Comdty"), "BID")

## Historical (last 30 days)
edb <- blpGetData(conn, "ED1 Comdty", "PX_LAST",
start=as.chron(Sys.time() - 86400 * 30))

## Intraday bars (last hour in 2 min bars)
edc <- blpGetData(conn, "ED1 Comdty", c("BID", "ASK"),
start=as.chron(Sys.time() - 3600), barfields="OPEN", barsize=2)

## Tick-by-tick (3 minutes starting an hour ago)
```

```
edd <- blpGetData(conn, "ED1 Comdty", c("BID"),
start=as.chron(Sys.time() - 3600),
end=as.chron(Sys.time() - 3420), barsize=0)

blpDisconnect(conn)
## End(Not run)
```

blpReadFields *Read bbfields.tbl file into workspace.*

Description

Read into the workspace the file `bbfields.tbl` located on the local machine.

Usage

```
blpReadFields(path = "C:/blp/API")
```

Arguments

`path` character. path where `bbfields.tbl` can be found.

Details

In order to perform datatype mapping, RBloomberg needs to consult the Bloomberg API file `bbfields.tbl` located on the machine running the Bloomberg terminal. We don't physically search the file each time a datatype mapping is needed. Instead, we read the entire contents of the file into the workspace and store it in a hidden dataframe `.bbfields`.

This is attempted when the package is first loaded. If the file is not found in `C:/blp/API`, you must manually specify the correct path using this function.

Author(s)

Robert Sams (robert@sanctumfi.com)

Examples

```
.bbfields <- blpReadFields()
```

Description

These functions are not to be called by the user. This is ad-hoc documentation for developers of this package.

Usage

```

blpGetHistoricalData(conn, securities, fields, start, end, barsize = NULL,
                    barfields = NULL)
blpSubscribe(conn, securities, fields)
replaceBloombergErrors(x, suppress = TRUE)
read.bbfields(path = "C:/blp/API")
dataType(mnemonic, bbfields = .bbfields)
## S3 method for class 'chron':
as.COMDate(x, date1904 = FALSE)
## S3 method for class 'COMDate':
as.chron(x, date1904 = FALSE, ...)
## S3 method for class 'BlpCOMReturn':
as.matrix(x, ...)
## S3 method for class 'BlpCOMReturn':
as.data.frame(x, row.names = NULL, optional =
FALSE, ...)
## S3 method for class 'BlpCOMReturn':
as.zoo(x, suppress = TRUE, bbfields = .bbfields, ...)

```

Arguments

x	object
conn	connection object
securities	vector of Bloomberg security codes
fields	vector of field mnumonics
start,end	chron object defining data window
barsize	see blpGetData documentaion
barfields	see blpGetData documentaion
date1904	see below
suppress	logical. do not raise warnings when Bloomberg error codes are converted to NA's
path	path to bbfields file user's workstation
mnemonic	Bloomberg field mnemonic, as documented in bbfields
bbfields	dataframe containing data in bbfields

row.names	ignored
optional	ignored
...	other args

Details

At the moment this package can connect to Bloomberg only via the COM Desktop interface. Support for other Bloomberg interfaces are envisioned, especially the C Server interface.

The user interacts with Bloomberg via the `blpGetData` function. This is a generic and a method will eventually exist for each Bloomberg interface. The idea is that calls to it and the values returned by it are exactly the same, regardless of the actual interface used, which is determined by the class of the connection object passed to the first argument of `blpGetData`. In short: user code is divorced from choice of interface. That's the aim. Currently, only one method exists, `blpGetData.BlpCOM`.

All COM requests return a nested list, the nesting structure implicitly defining the return value's dimension and preserving data types. At present, we only support two-dimensional return values (e.g., a multiple security, multiple field timeseries request is not allowed).

First, the nested list is transformed into something useful by coercing it to a vector, thereby implicitly coercing everything to the character data type if there is at least one non-numeric item in the return. If there are any Bloomberg error codes, these are then converted to NA's. Finally, the columns are recovered by setting the `dim` attribute based on what we know about the request. If the request is for a timeseries, the leftmost column is the COM date index, represented numerically. We use `as.chron(mtx[1,])` to get back chron dates.

If there are multiple data types in a single return, a dataframe is the natural structure for Bloomberg return values. But how do we recover the data types? (Note: a single Bloomberg error code in a numeric field will cause that field to be coerced into character mode on the strategy implemented here.) When `RBloomberg` is attached, the `.bbfields` file on the user's workstation is read and stored as a dataframe in `.bbfields` on the workspace. Every field available from the Bloomberg API is mapped onto a datatype in this file. `RBloomberg` maps these datatypes to R datatypes. We use this information to recover datatypes when a dataframe is required.

`blpGetHistoricalData` and `blpSubscribe` are both wrappers to the COM methods of the same name. Depending on what is passed to `blpGetHistoricalData`, either a historical, intraday bars, or intraday tick request is made. `blpSubscribe` is for non-time series requests. Both functions return an object of class `'BlpCOMReturn'`, which is nothing more than the list returned by the COM method with some attributes attached. The attributes provide meta-information that the coercion methods need to coerce the nested list returned by COM.

Users pass chron dates to `blpGetData` and they always get chron dates. But COM requests expect COM dates and return COM dates, which are floating point reals representing the number of days since 1900. Two coercion functions `as.Date.comDate` and `as.comDate.Date` are provided to deal with this. For a discussion of the issue surrounding the treatment of 1904, see <http://cpearson.com/excel/datetime.htm>.

Author(s)

Robert Sams (robert@sanctumfi.com)

References

From your Bloomberg terminal, use function WAPI:

"BLPGetHistoricalData Property"	Ref# FRP003
"History, Raw Tick and Time-Bar Requests"	Ref# DP015
"Extended Historical Data Requests"	Ref# DBExt
"Appendix C: Ticker Syntax"	Ref# APX003
"Appendix D: Reading the Data Dictionary"	Ref# CAX041
"Appendix G: Error Codes"	Ref# APX007

Also, threads in the R-sig-finance mailing list archives:

"Import from Bloomberg"	Enrique Bengoechea, 6 Jul 04
"Bloomberg data import"	David L. Reiner, 31 Mar 05

Index

*Topic **math**

- blpConnect, 1
- blpGetData, 2
- blpReadFields, 4
- lowlevel, 5

- as.chron.COMDate (*lowlevel*), 5
- as.COMDate.chron (*lowlevel*), 5
- as.data.frame.BlpCOMReturn
 (*lowlevel*), 5
- as.matrix.BlpCOMReturn
 (*lowlevel*), 5
- as.zoo.BlpCOMReturn (*lowlevel*), 5

- blpConnect, 1
- blpDisconnect (*blpConnect*), 1
- blpGetData, 2
- blpGetHistoricalData (*lowlevel*), 5
- blpReadFields, 4
- blpSubscribe (*lowlevel*), 5

- dataType (*lowlevel*), 5

- lowlevel, 5

- read.bbfields (*lowlevel*), 5
- replaceBloombergErrors
 (*lowlevel*), 5