BloomR main functions

R topics documented:

br.bdh

Historical data

Description

Download historical Bloomberg data

Usage

```
br.bdh(con, securities, fields="PX_LAST", start.date, end.date = NULL,
option.names = NULL, option.values = NULL,
always.display.tickers = FALSE, dates.as.row.names = (length(securities) == 1),
include.non.trading.days = NULL
)
```

Arguments

con the connection token returned from br.open()
securities character vector of the tickers queried for data
field case insensitive character vector of the Bloomberg field queried. Defaults to "PX_LAST".
start.date Time series start date as a Date object or an ISO string without separators (YYYYMMDD). Time series will actually begin at `start.date` if there is data available; otherwise it will start at the first significant date.
end.date Time series end date as a Date object or an ISO string without separators (YYYYMMDD). If NULL or missing, it defaults to the last available date.
option.names, option.values See details
always.display.tickers Displays tickers in first column even if a single one is requested. Defaults to FALSE
dates.as.row.names Displays dates *also* as row names. Defaults to TRUE for single ticker query, FALSE otherwise.

include.non.trading.days If TRUE, rows are returned for all requested dates, even when markets no data is available. It defaults to FALSE for a single security or and TRUE for multiple securities. In the latter case, use `na.omit` or `na.exclude` to remove these rows.

Details

For multi-ticker queries you might consider to use `br.bulk.tiks` which features also a simulated mode.

`option.names` and `option.values` are options vectors affecting the returned data. Options are set pairwise, for example to set `opt1` and `opt2` respectively to `val1` and `val2`, you would pass the arguments:

```
option.names=c("opt1", "opt2"), option.values=c("val1", "val2")
```

Here is a list of options:

periodicityAdjustment Determine the frequency and calendar type of the output. To be used in conjunction with `periodicitySelection`. If `ACTUAL`, it reverts to the actual date from today (if the end date is left blank) or from the End Date. If `CALENDAR`, (for pricing fields), it reverts to the last business day of the specified calendar period. Calendar Quarterly (CQ), Calendar Semi-Annually (CS), or Calendar Yearly (CY). If `FISCAL`, it reverts to the fiscal period end for the company - Fiscal Quarterly (FQ), Fiscal Semi-Annually (FS) and Fiscal Yearly (FY) only.

periodicitySelection Determine the frequency of the output. To be used in conjunction with periodicityAdjustment. if `DAILY`, `WEEKLY`, `MONTHLY`, `QUARTERLY`, `SEMI_ANNUALLY`, `YEARLY`.

currency Amends the value from local to desired currency. The value is a 3 letter ISO code string, e.g. USD, GBP. View `WCV<GO>` on the Bloomberg terminal for the full list.

overrideOption Indicates whether to use the average or the closing price in quote calculation. Values can be `OVERRIDE_OPTION_CLOSE` for using the closing price or `OVERRIDE_OPTION_GPA` for the average price.

pricingOption Sets quote to Price or Yield for a debt instrument whose default value is quoted in yield (depending on pricing source). `PRICING_OPTION_PRICE` sets quote to price; `PRICING_OPTION_YIELD` sets quote to yield.

nonTradingDayFillOption Sets to include/exclude non trading days where no data was generated. `NON_TRADING_WEEKDAYS` includes all weekdays (Monday to Friday); `ALL_CALENDAR_DAYS` includes all days of the calendar; `ACTIVE_DAYS_ONLY` includes only the days where the instrument and field pair were updated.

nonTradingDayFillMethod If data is to be displayed for non trading days what is the data to be returned. `PREVIOUS_VALUE` searches back and retrieve the previous value available for this security field pair. The search back period is up to one month. `NIL_VALUE` returns blank for the "value" value within the data element for this field.

maxDataPoints the maximum number of data points to return. If the original data set is larger, the response will be a subset, containing only the last `maxDataPoints` data points.

returnEids returns the entitlement identifiers associated with security. If TRUE, populates data with an extra element containing a name and value for the EID date.

returnRelativeDate returns data with a relative date. If TRUE, populates data with an extra element containing a name and value for the relative date. For example RELATIVE_DATE = 2002 Q2

adjustmentNormal Adjust for "change on day". If TRUE, adjusts historical pricing to reflect: Regular Cash, Interim, 1st Interim, 2nd Interim, 3rd Interim, 4th Interim, 5th Interim, Income, Estimated, Partnership Distribution, Final, Interest on Capital, Distribution, Prorated.

adjustmentAbnormal Adjusts for Anormal Cash Dividends. If TRUE, adjusts historical pricing to reflect: Special Cash, Liquidation, Capital Gains, Long-Term Capital Gains, Short-Term Capital Gains, Memorial, Return of Capital, Rights Redemption, Miscellaneous, Return Premium, Preferred Rights Redemption, Proceeds/Rights, Proceeds/Shares, Proceeds/Warrants.

adjustmentSplit Capital Changes Defaults. If TRUE, adjusts historical pricing and/or volume to reflect: Spin-Offs, Stock Splits/Consolidations, Stock Dividend/Bonus, Rights Offerings/Entitlement.

adjustmentFollowDPDF If TRUE (defaults) Follow the Bloomberg function as from `DPDF<GO>`.

CalendarCodeOverride Returns the data based on the calendar of the specified country, exchange, or religion. Value is a two character calendar code as from `CDR<GO>`. This will cause the data to be aligned according to the calendar and including calendar holidays. Only applies only to DAILY requests.

calendarOverridesInfo (Experimental, not tested) Returns data based on the calendar code of multiple countries, exchanges, or religious calendars as from `CDR<GO>`. This will cause the data to be aligned according to the set calendar(s) including their calendar holidays and only applies to DAILY requests. Requires `calendarOverrides`, which is a character vector of two-character calendar codes as from `CDR<GO>`; `calendareOverridesOperation`, which can be `CDR_AND` returning the intersection of trading days among multiple calendars or `CDR_OR` returning the union of trading days. That is, a data point is returned if a date is a valid trading day for any of the calendar codes specified in the request.

Overrides (Experimental, not tested) Append overrides to modify the calculation. `fieldID` specifies a field mnemonic or alpha-numeric, such as `PR092` or `PRICING_SOURCE`. Review FLDS for list of possible overrides. `value` sets the desired override value

Value

A data frame with historical data. If tickers are displayed, the first column shows tickers, the second one the time series dates and the following ones the values of the queried fields; otherwise the columns start with dates. Dates will also be shown as rows if `dates.as.row.names=TRUE`. If multiple tickers are queried, they are vertically stacked respecting the order in `securities` vector.

br.bulk.csv

Historical data from grouped tickers in a CSV files
Reads a CSV file containing a group of tickers in each column and returns the historical data in xts or list format. The CSV file is assumed to have headers denoting group labels.

Usage

```
br.bulk.csv(con, file, start = Sys.Date() - 5, field = "PX_LAST",
    cols = NULL, addtype = FALSE, showtype = FALSE, use.xts = TRUE,
    comma = TRUE,
    price=TRUE, nrow=5, same.dates=FALSE, no.na=FALSE, empty.sec=0
    )
```

Arguments

con the connection token returned from br.open(). If `NULL` simulated values are generated.

file path to CSV file.

start start date. Can be a Date object or an ISO string without separators (YYYYMMDD). Defaults to 5 days before current date.

field case insensitive string denoting the Bloomberg field queried. Defaults to "PX_LAST". If the field is wrong or not accessible, data will be empty but no error will be raised.

cols Logical or integer vector for selecting CSV columns (ticker groups). Defaults to all columns.

addtype If a string denoting the security type, it will be added to all tickers; if TRUE "Equity", will be added; if FALSE (the default), nothing will be added.

showtype if TRUE, security types will be removed from names of list or xts output. It defaults to FALSE.

use.xts if TRUE (the default) each group will be formatted as an xts object else as a list.

comma to be set to FALSE for (non-English) CSV, using semicolon as separator.

nrow maximum number of simulated rows (actual is random). Ignored if `con!=NULL`, it defaults to 5.

empty.sec ratio of securities returning no data. Ignored if `con!=NULL`, it defaults to 0.

Details

Empty CSV cells or cells interpreted as NAs will be ignored.
If `con=NULL`, values are simulated by means of `br.sample()`. This function is used with default values, except for `nrow, start, same.dates, no.na, empty.sec`, which can be explicitly passed as arguments, and `sec.names` depending on tickers found in the CSV file. These arguments are ignored if `con!=NULL`. See `br.sample()` help for more.

Value

a list where each element is the historical data of a CSV group.
If `use.xts=TRUE`, elements are xts object, where each column is the historical data of a security.
If `use.xts=FALSE`, elements are sub-list, where each element is the historical data of a security.
If there is only one group, the first (and unique) element of the list will be returned.

Demonstration

A sample CSV with Bloomberg tickers will look like follows:

```
read.csv("mybloomr/tickers.csv")
## This file is part of BloomR and anyway available here:
## https://github.com/AntonioFasano/BloomR/blob/master/res/tickers.csv
```

```
##          Financial     Technology      Indices
## 1   3988 HK Equity QCOM US Equity    DJI Index
## 2      C US Equity CSCO US Equity DJUSFN Index
## 3 601288 CH Equity  700 HK Equity  W1TEC Index
## 4    BAC US Equity  IBM US Equity
## 5   HSBA LN Equity INTC US Equity
```

Note:

- CSV group headers are mandatory;
- Group headers need not to be the same length.

We can now download data:

```
con=NULL  #Simulated mode: replace with con=br.open() on terminal
```

```
data=br.bulk.csv(con, "mybloomr/tickers.csv")
```

```
## Processing Financial ...
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading 3988 HK Equity
```

```
## Loading C US Equity
```

```
## Loading 601288 CH Equity
```

```
## Loading BAC US Equity
## Loading HSBA LN Equity
## Processing Technology ...
## Loading QCOM US Equity
## Loading CSCO US Equity
## Loading 700 HK Equity
## Loading IBM US Equity
## Loading INTC US Equity
## Processing Indices ...
## Loading DJI Index
## Loading DJUSFN Index
## Loading W1TEC Index
```

Above you see some info about data being processed that we will not show anymore in the following.

If you want to have detailed ticker descriptions, see br.bulk.desc Example. Downloaded data look like follows:

```
data
```

```
## $Financial
##            3988 HK   C US 601288 CH BAC US HSBA LN
## 2016-10-24      NA 10.797    11.018     NA   9.852
## 2016-10-25   9.874 12.121    11.369 11.156   9.638
## 2016-10-26   9.720 10.556        NA     NA   9.636
## 2016-10-27   9.115  9.353        NA  8.869  10.154
## 2016-10-28  10.262 10.297        NA     NA   8.592
##
## $Technology
##            QCOM US CSCO US 700 HK IBM US INTC US
## 2016-10-24   9.896   9.652 11.751     NA      NA
## 2016-10-25   9.380  10.255  9.724     NA   9.443
## 2016-10-26  10.034   8.270  9.987     NA      NA
## 2016-10-27   9.038  11.078  9.361     NA      NA
## 2016-10-28      NA  11.048     NA  10.95      NA
##
## $Indices
##              DJI DJUSFN W1TEC
## 2016-10-24 9.992 10.719 9.082
## 2016-10-25    NA     NA 9.639
## 2016-10-26    NA  8.994    NA
## 2016-10-27 9.584  8.949 8.995
## 2016-10-28 9.299 10.947 7.887
```

Note:

- The name of the securities tickers is stored without the security type: “Equity”, “Index”, etc.
  If this piece of info is significant for you, pass `showtype = TRUE`.
- Time series start date defaults to 5 days before current date, unless you set `start` to: an R Date object (`start=as.Date("2014/9/30")`) or to a more friendly ISO string (`start="20140930"`).

Data are stored as a list of xts objects, each representing one group of tickers in the CSV file.

```
length(data)
```

```
## [1] 3
```

```
names(data)
```

```
## [1] "Financial"  "Technology" "Indices"
```

```
class(data$Financial)
```

```
## [1] "xts" "zoo"
```

If you prefer you may get time series as data frames, and precisely as a list representing the ticker groups, where each group is in turn a list containing a data frame for each security:

```
data=br.bulk.csv(con, "mybloomr/tickers.csv", use.xts=FALSE)
```

```
length(data)
```

```
## [1] 3
```

```
names(data)
```

```
## [1] "Financial"  "Technology" "Indices"
```

```
class(data$Financial)
```

```
## [1] "list"
```

```
length(data$Financial)
```

```
## [1] 5
```

```
names(data$Financial)
```

```
## [1] "3988 HK"   "C US"      "601288 CH" "BAC US"    "HSBA LN"
```

```
class(data$Financial$`BAC US`)
```

```
## [1] "data.frame"
```

By defaults time series list values from the Bloomberg "PX_LAST" field. To change the default field use:

```
data=br.bulk.csv(con, "mybloomr/tickers.csv", field = "PX_OPEN")
```

You can choose to import only some of the CSV groups

```
## Processing Financial ...
## Loading 3988 HK Equity
## Loading C US Equity
## Loading 601288 CH Equity
## Loading BAC US Equity
## Loading HSBA LN Equity
## Processing Indices ...
## Loading DJI Index
## Loading DJUSFN Index
## Loading W1TEC Index
```

```
data=br.bulk.csv(con, "mybloomr/tickers.csv", cols=c(1,3))
## or equivalently:
data=br.bulk.csv(con, "mybloomr/tickers.csv", cols=c(TRUE, FALSE, TRUE))
```

```
names(data)
```

```
## [1] "Financial" "Indices"
```

In the CSV file, if your tickers represent all equities, you can omit the type.

Consider this CSV:

```
read.csv("mybloomr/tickers.eqt.csv")
## This file is part of BloomR and anyway available here:
## https://github.com/AntonioFasano/BloomR/blob/master/res/tickers.eqt.csv
```

```
##   Financial Technology
## 1   3988 HK    QCOM US
## 2      C US    CSCO US
## 3 601288 CH     700 HK
## 4    BAC US     IBM US
## 5   HSBA LN    INTC US
```

Note how the "Equity" type is missing! But you can use this CSV file with `addtype`:

```
data=br.bulk.csv(con, "mybloomr/tickers.eqt.csv", addtype=TRUE)
```

Before going home, don't forget to:

```
br.close(con)
```

br.bulk.desc

Description

Get security descriptions for a vector of tickers.

Usage

```
br.bulk.desc(con, tiks)
```

Arguments

con the connection token returned from br.open()
tiks character vector of the tickers queried for data

Value

A list of data frames, each representing the description of a security. For the format of data frames see the function `br.desc`.

Example

```
con=br.open()
data=read.csv("mybloomr/tickers.csv", as.is=TRUE)
br.bulk.desc(con, as.vector(as.matrix(data[1:2,])))
br.close(con)
```

br.bulk.idx

Description

Returns the historical data for the constituents of an index in xts or list format.

Usage

```
br.bulk.idx(con, index, start = Sys.Date() - 5, field = "PX_LAST",
     showtype = FALSE, include.idx = TRUE, use.xts = TRUE, nsec = 50,
     price = TRUE, nrow = 5, same.dates=FALSE, no.na=FALSE, empty.sec = 0,
     sec.names = NULL)
```

Arguments

con the connection token returned from br.open(). If `NULL` simulated values are generated.
index string denoting the index ticker with or without the final security type label ('Index')
include.idx if TRUE (default) returns also historical data for the index.
nsec number of simulated index constituents. Ignored if `con!=NULL`, it defaults to 10.
sec.names character vector with names of sampled index constituents. Ignored if `con!=NULL`. By default security names are like 'memb1', 'memb2', etc.

For other arguments see the function `br.bulk.csv`

Details

If `con=NULL`, values are simulated by means of `br.sample()`. This function is used with default values, except for `nrow, nsec1, price, start, same.dates, no.na, empty.sec, sec.names`.

Value

If `use.xts=TRUE`, an xts object, where each column is the historical data of a constituent.
If `use.xts=FALSE`, a list, where each element is the historical data of a constituent.
If `include.idx=TRUE`, the last column or element will be the historical data of the index.

br.bulk.tiks

Bulk historical data
Returns the historical data for a vector of tickers in xts or list format

Usage

```
br.bulk.tiks(con, tiks, start=Sys.Date()-5, field="PX_LAST",
    addtype=FALSE, showtype=FALSE, use.xts=TRUE,
    price=TRUE, nrow=5, same.dates=FALSE, no.na=FALSE, empty.sec=0)
```

Arguments

tiks character vector of the tickers queried for data

For other arguments see the function `br.bulk.csv`

Details

If an element of `tiks` is `NA` or empty (`""`) it is ignored. This is intended to avoid errors when the character vector are read from a CSV file with empty cells.
If `con=NULL`, values are simulated by means of `br.sample()`. Sampled values are based on default values of `br.sample()`, but it is possible to set explicitly `start, same.dates, no.na, empty.sec`; `sec.names` depends on `tiks` argument. These arguments are ignored if `con!=NULL`. See `br.sample()` help for more.

Value

If `use.xts=TRUE`, an xts object, where each column is the historical data of a security.
If `use.xts=FALSE`, a list, where each element is the historical data of a security.

Example

```
con=NULL # Open simulated connection and  load some data
br.bulk.tiks(con, c("MSFT US", "AMZN US"), addtype=TRUE)
```

```
## Loading MSFT US Equity
```

```
## Loading AMZN US Equity
```

```
##            MSFT US AMZN US
## 2016-10-24      NA  10.541
## 2016-10-25   9.883  10.326
## 2016-10-27   9.505  10.499
## 2016-10-28  10.838  10.080
```

```
br.close(con) # Use the token to release the connection
```

See Also

br.bulk.csv

br.desc

Description

Get security descriptions.

Usage

```
br.desc(con, tik)
```

Arguments

con the connection token returned from br.open()
tik string denoting the ticker queried for data

Value

A data frame containing the value of the Bloomberg fields form `ds001` to `ds009` and the long field `CIE_DES_BULK`.

br.md2pdf

Description

Make a markdown file into a PDF It assumes that you have installed the BloomR LaTeX addons

Usage

```
br.md2pdf(md.file, pdf.file)
```

Arguments

md.file path to the markdown file to be converted.
pdf.file path to the PDF file to be generated.

Details

The function will stop with an error if you have not installed BloomR LaTeX addons. To install them use `br.getLatexAddons()`.

Value

If there are no errors, it returns zero invisibly, otherwise it prints an error message and returns the related error code.

br.sample

Description

Return simulated historical data for n securities in xts or df format.

Usage

```
br.sample(nrow, nsec=1, price=TRUE, start=Sys.Date(),
mean=ifelse(price, 10, 0.1), sd=1, jitter=0, same.dates=FALSE, no.na=FALSE,
empty.sec=0, df=FALSE, sec.names=NULL)
```

Arguments

nrow number of simulated data points for each security; if `same.dates=FALSE`, the number of rows for each sampled security will be a random number not exceeding nrow, else it will be nrow for all securities.
nsec number of simulated securities (defaults to 1).
price if TRUE (default), simulated values are non-negative.
start start date. Can be a Date object or an ISO string without separators (YYYYMMDD). Defaults to current date.
mean mean of security generated values. If `price=TRUE`, default to 10 else defaults to 0.1.
sd sd of security generated values. It defaults to 1.
jitter modifies each security mean by adding adding a random value in [-jitter, jitter]. Defaults to 0.
same.dates if TRUE, all sampled securities will refer to the same dates and for each security the number will equal nrow. If FALSE (default), date values and number will randomly differ. For each security the random number will not exceed `nrow`.
no.na if `same.dates=FALSE`, when merging sampled security data NAs are likely to be produced. If `no.na=FALSE` (default) they will be left, otherwise they will be removed using R `na.omit`
df if FALSE (default), the output will be an xts object, else the output will be a data frame with the first column containing the dates of the sampled data.
sec.names character vector for column names. If `df=FALSE` the length of the vector should be equal to `nsec`, else to `nsec + 1` (because of the first column containing dates). By default security names are like 'sample1', 'sample2', etc. and the date column is named 'date'.
empty.sec ratio of securities returning no data (defaults to 0). The result is rounded without decimal places.

Value

If `df=TRUE`, a data frame object, where the first column is the vector with all generated dates merged and each subsequent column contains the sampled data of a security. If `df=FALSE`, an xts object, where each element is the sampled data of a security, while the dates will be part of the xts time object. In both cases if `same.dates=FALSE` and/or `empty.sec!=0` generated data points will have different length and the the date gaps will be filled with NAs, except if `no.na=TRUE`. If the generated values are only NAs the output will be converted to a 0-rows xts or data frame, containing only security labels accessible with `dimnames(*)[[2]]`.

Deprecated functions

Description

Functions not used anymore generating an informative error

Usage

```
bbg.open()
bbg.close(con)
```

Arguments

con the connection token returned from br.open()

Example

```
con=bbg.open()
##  Sorry 'bbg.open' is now deprecated. Please use br.open().
```

Internal BloomR functions

Description:

Internal functions not to be used by the end user

Usage:

```
.br.is.con(con)
.br.types
.br.check.type(type)
.br.cuttype(type)
.br.jar()
```

Arguments:

con the connection token returned from br.open()
type a string representing the security type

Details

`.br.is.con` checks for the validity of a connection token. `.br.types` is a character vector with security types suitable as an argument for `br.bulk*` functions. `.br.check.type` checks if a type matches `.br.types`. `.br.cuttype` cuts trailing security type from character vector. `.br.jar()` returns the path to the blpapi*.jar

Manage connections

Description

Open and close the connection to the Bloomberg service.

Usage

```
br.open()
br.close(con)
```

Arguments

con the connection token returned from br.open()

Details

`br.open` returns the connection token needed by the BloomR function downloading data. When you finish you session, you pass it to `br.close`. If you are using simulated data and so your connection token is NULL, closing the connection is optional. Anyway running `br.close(con)`, even if `con==NULL` avoids adding this line when you switch to a actual data download.

Example

```
con=br.open() # Open the connection and get the token and load some data
br.bulk.tiks(con, c("MSFT US", "AMZN US"), addtype=TRUE)
br.close(con) # Use the token to release the connection
```

Misc functions

Description

`rm.all` deletes all objects (variables and functions) from memory, including invisible objects (those starting with a dot). `rm.var` deletes non-function objects from memory.

Usage

```
rm.all()
rm.var()
```

Time extension functions

Description

Functions to get, set dates.

Usage

```
day(d)
month(d)
year(d)
```

```
day(d, n)
month(d, n)
year(d, n)
day(d)=x
month(d)=x
year(d)=x
d %+% n
d %-% n
last.day(d)
day.us(d1, d2)
```

Arguments

d, d1, d2 objects of class date
x an integer representing the day/month/year
n an integer representing the months to add/subtract

Details

If `component` is `day`, `month` or `year`: `component(d)` returns the *component* of the date `d` as an integer; `component(d, n)` returns the date `d` with the *component* set to the integer `n`; `component(d)= n` sets to the *component* of the date `d` to the integer `n`.
`%+%` and `%-%` add and subtract months to a date.
`last.day` returns last day of the month as an integer. `day.us` calculates date differences with the US convention.