Package ‘DatastreamDSWS2R’
November 25, 2020

Type Package
Title Provides a Link Between the 'Refinitiv Datastream' System and R
Version 1.7.9
Date 2020-11-23
Author Charles Cara
Maintainer Charles Cara <charles.cara@absolute-strategy.com>
Description Provides a set of functions and a class to connect, extract and upload information from the 'Refinitiv Datastream' database. This package uses the 'DSWS' API and server used by the 'Datastream DFO addin'. Details of this API on the 'DSWS' page of <https://developers.refinitiv.com/en>. Please report issues at <https://github.com/CharlesCara/DatastreamDSWS2R/issues>.
License GPL-3
LazyData TRUE
Imports httr, jsonlite, stringi, stringr, xts, zoo, methods, foreach, dplyr
Suggests testthat
RoxygenNote 7.1.1
Collate 'DatastreamDSWS2R.R' 'common.R' 'classConstructor.R'
            'wrapper.R' 'UCTSUpload.R' 'cbindRobust.R' 'data.R'
Encoding UTF-8
Depends R (>= 2.10)
NeedsCompilation no
Repository CRAN
Date/Publication 2020-11-25 14:20:04 UTC

R topics documented:

  cbindRobust ............................................................. 2
  classconstructor ........................................................ 2
  currencyDS2ISO .......................................................... 7
cbindRobust

Function to combine time series that fixes the NA problem

Description

When combining two xts time series in which one series is an empty NA series and the other is a character series, then the normal cbind function will return a time series with the correct number of rows and columns but with every cell occupied with NA. This function overcomes this problem by allowing us to combine an empty series and a character series.

Usage

`cbindRobust(xts1, xts2)`

Arguments

- `xts1` First time series to combine
- `xts2` Second time series to combine

classconstructor

dsws

Description

An R5/RC object for accessing the Refinitiv Datastream DSWS service.

Details

Creates an R5/RC4 object for accessing the Refinitiv Datastream DSWS service.
Fields

tokenList fieldDescription
tokenSource fieldDescription
serverURL fieldDescription
username fieldDescription
password fieldDescription
initialised fieldDescription
errorList fieldDescription
requestList fieldDescription
jsonResponseSaveFile fieldDescription
jsonResponseLoadFile fieldDescription
dataResponse fieldDescription
symbolList fieldDescription
myValues fieldDescription
myTypes fieldDescription
logging fieldDescription
numDatatype fieldDescription
numInstrument fieldDescription
chunkLimit fieldDescription
requestStringLimit fieldDescription
logFileFolder fieldDescription

Methods

initialize( dws.serverURL = "", getTokenFunction = NULL, token = NULL, username = "", password = "", connect = TRUE )
initialises the class. Unless noConnect is TRUE also connects to the Datastream dws server.
Authentication can be set in three ways: 1) If getTokenFunction is not null then that function
is called. It is expected to return a list with items 'TokenValue' and 'TokenExpiry'.
2) An access token can also be passed into the class on initialisation, so that it can be shared
between sessions. 'token' is expected to be a list with items 'TokenValue' and 'TokenExpiry'.
3) A username and password that are used to fetch a token from the DSWS server. If the user-
name and password are not provided, then they are sourced from system environment variables
(ie Sys.getenv) 'DataStreamUsername' and 'DataStreamPassword' or alternatively (not pre-
ferred) that from options()$Datastream.Username and options()$Datastream.Password
This allows the password to be stored in .Renviron or .RProfile rather than in the source code.
There is a difference in the Refinitiv's documentation about the chunk limit and different
accounts have different limits. Some users are limited to 50 items while others are limited to
2000L. The chunk limit can be controlled by setting the chunkLimit parameter of the dws
object. If options()$Datastream.ChunkLimit is set then the value is taken from there.

listRequest(instrument, datatype = "", expression = "", requestDate)
Make a listRequest
from Datastream DSWS. This is the equivalent to the Excel static request for a list.
Parameters are:
**instrument** should contain a list mnemonic, such as 'LFTSE100'. Can be a user created list or index. The UCL can contain expressions.

**datatype** array of datatypes eg NAME, MNEM, P, PE etc

**expression** if datatype is null or ' then an expression eg PCH#(XXXX,3M)

**requestDate** either a Date or a string with a datastream relative date eg '-3M'

Returns a data.frame with the requested data.

Examples:

```r
mydsws$listRequest(instrument = "LFTSE100",
                     datatype = c("NAME","P"),
                     requestDate = "-0D")
```

```r
mydsws$listRequest(instrument = "LFTSE100",
                     expression = "PCH#(XXXX,3M)", requestDate = Sys.Date())
```

**snapshotRequest(instrument, datatype = "", expression = "", requestDate)** Make a snapshotRequest from Datastream DSWS. This is the equivalent to the Excel static request for an array of instruments.

Parameters are:

**instrument** should one or more instruments eg "MKS" or c("MKS","@AAPL"). The array can contain Economics codes and Expressions.

**datatype** array of datatypes eg NAME, MNEM, P, PE etc

**expression** if datatype is null or " then an expression eg PCH#(XXXX,3M)

**requestDate** either a Date or a string with a datastream relative date eg '-3M'

Returns a data.frame with the requested data.

Examples:

```r
mydsws$snapshotRequest(instrument = c("MKS","@AAPL"),
                        datatype = c("NAME","P"), requestDate = "-0D")
```

```r
mydsws$snapshotRequest(instrument = c("MKS","@AAPL"),
                        expression = "PCH#(XXXX,3M)", requestDate = "-0D")
```

**timeSeriesListRequest( instrument, datatype = "", expression = "", startDate, endDate, frequency = "D", format = "ByInstrument" )** Make a timeSeriesListRequest from Datastream DSWS. This is the equivalent to the Excel timeseries request for an array of instruments. Should request either a datatype or an expression not both. If a datatype is provided then anything in Expression will be ignored.

Parameters are:

**instrument** should contain a list mnemonic, such as "LFTSE100". Can be a user created list or index. The UCL can contain expressions.

**datatype** array of datatypes eg P, PE etc
expression if datatype is null or " then an expression
e.g. PCH#(XXXX,3M)

startDate either a Date or a string with a datastream relative date
e.g. '-3M'

endDate either a Date or a string with a datastream relative date
e.g. '-0D'

frequency one of the standard Datastream frequencies - D, W, M, Q, or Y

format can be either "ByInstrument" or "ByDatatype".

Returns either a single xts or a list of xts a data.frame with the requested data. If "ByInstrument" then the data is returned as one or more (ie a list) wide xts with one column per instrument. If "ByDatatype" then the data is returned as one or more (ie a list) of wide xts with one column per Datatype. This format is more compatible with the quantmod package.

Examples:

```r
mydsws$timeSeriesListRequest(instrument = "LFTSE100",
   datatype = "P", startDate = "-30D",
   endDate = "-0D", frequency = "D")
```

```r
mydsws$timeSeriesListRequest(instrument = "LFTSE100",
   expression = "PCH#(XXXX,3M)",
   startDate = "-30D",
   endDate = "-0D", frequency = "D")
```

```r
mydsws$timeSeriesListRequest(instrument = "LFTSE100",
   datatype = ("P","UP"), startDate = "-30D",
   endDate = "-0D",
   frequency = "D", format = "ByDatatype")
```

timeSeriesRequest( instrument, datatype = "", expression = "", startDate, endDate, frequency = "D", format = "ByDatatype")

Return a timeSeriesRequest from Datastream dsws. Should request either a datatype or an expression not both. If a datatype is provided then anything in Expression will be ignored.

Make a timeSeriesRequest from Datastream DSWS. This is the equivalent to the Excel time-series request for an array of instruments.

Parameters are:

instrument should one or more instruments eg "MKS" or c("MKS","@AAPL"). The array can contain Economics codes and Expressions.
datatype array of datatypes eg P, PE etc
expression if datatype is null or " then an expression eg PCH#(XXXX,3M)
startDate either a Date or a string with a datastream relative date eg '-3M'
endDate either a Date or a string with a datastream relative date eg '-0D'
frequency one of the standard Datastream frequencies - D, W, M, Q, or Y
format can be either "ByInstrument" or "ByDatatype".

Returns either a single xts or a list of xts a data.frame with the requested data. If "ByInstrument" then the data is returned as one or more (ie a list) wide xts with one column per instrument. If "ByDatatype" then the data is returned as one or more (ie a list) of wide xts with one column per Datatype. This format is more compatible with the quantmod package.

Examples:

```r
mydsws$timeSeriesRequest(instrument = c("MKS","@AAPL"),
                       datatype = "P", startDate = "-30D",
                       endDate = "-0D", frequency = "D")
```

```r
mydsws$timeSeriesRequest(instrument = c("MKS"),
                       expression = "PCH#(XXXX,3M)", startDate = "-30D",
                       endDate = "-0D", frequency = "D")
```

```r
mydsws$timeSeriesRequest(instrument = c("MKS","@AAPL"),
                       datatype = ("P","UP"), startDate = "-30D",
                       endDate = "-0D", frequency = "D", format = "ByDatatype")
```

Examples

```r
## Not run:
mydsws <- dsws$new()
# Snapshot requests
myData <- mydsws$snapshotRequest(instrument = c("ABF","RIO","WPP"),
                                   datatype = "P",
                                   requestDate = "0D")
myData <- mydsws$snapshotRequest(instrument = c("ABF","RIO","WPP"),
                                   expression = "PCH#(XXXX,3M)",
                                   requestDate = "0D")
```
```r
myData <- mydsws$listRequest(instrument = "LFTSE100", datatype = "P", requestDate = "0D")

mydsws$snapshotRequest(instrument = c("SWCNB10","UKEUSCCIR"),
    datatype = c("MNEM","UPDATE"),
    requestDate = "0D")

mydsws$snapshotRequest(instrument = c("VOD", "HSBA"),
    datatype = "QTEALL",
    requestDate = Sys.Date())

mydsws$snapshotRequest(instrument = "STATS",
    datatype = "DS.USERSTATS",
    requestDate = Sys.Date())

# Timeseries requests

xtsData <- mydsws$timeSeriesRequest(instrument = "MKS",
    datatype = "MV",
    startDate = "-30D",
    endDate = "-0D",
    frequency = "D")

xtsData <- mydsws$timeSeriesListRequest(instrument = "LFTSE100",
    datatype = "MV",
    startDate = "-30D",
    endDate = "-0D",
    frequency = "D")

## End(Not run)
```

---

**currencyDS2ISO**

*Conversion table of Datastream to ISO currency codes*

**Description**

Conversion table of Datastream to ISO currency codes

**Usage**

currencyDS2ISO

**Format**

A data frame with 161 rows and 3 variables:

- **dsCode** the datastream code
- **isoCode** the ISO code for the currency
- **primeCode** primaryCode for currency or alternative
- **Multiplier** the units of the currency
Description

A package to manage access to the Refinitiv Datastream DSWS webservice

Usage

ggetDataStream(
    dweURLwsdl = "",
    User = as.character("USERNAME"),
    Pass = as.character("PASSWORD")
)

Arguments

dweURLwsdl Ignored
User Ignored - now sourced from options()$Datastream.Username
Pass Ignored - now sourced from options()$Datastream.Password

Details

Initialise connection with Datastream DSWS server. Provided for backwards compatibility

Value

a dsws object
listRequest  

Make a list request for static data (Deprecated)

Description

listRequest Function that returns a the value of Expression for the instrument list in DSCode from Datastream

Usage

listRequest(
  dwei = getDataStream(),
  DSCode,
  Expression = "",
  startDate = Sys.Date(),
  endDate = Sys.Date(),
  frequency = "D",
  verbose = FALSE
)

Arguments

dwei - A Datastream Client Interface object created with getDataStream
DSCode - the constituent list for the request eg LDJSTOXX
Expression - the data to return eg MNEM or NAME. If NULL or "" then we will return the code that has been loaded into the User Created List.
startDate - the date of the request, or the string "TODAY"
endDate - Ignored
frequency - the frequency of the request
verbose - whether to give messages during the request

Details

Make a list request for static data

Value

returns an array of the requested information
myStaticRequestSet  myStaticRequestSet (Depreciated)

Description

internal function for requesting an expression for an array of instruments. The function will initially try a snapshot request, and if this fails try a timeseries request.

Usage

myStaticRequestSet(
  mydsws = dsws$new(),
  instrument,
  iExpression,
  endDate = Sys.Date(),
  frequency = "D"
)

Arguments

mydsws  a dsws object, if not provided a new one will be created
instrument  array of instruments
iExpression  an expression such as PCH#(XXXX,1M)
endDate  the date of the request
frequency  optional frequency defaults to "D"

Details

Internal function

Value

a dataframe of the

staticListRequestSet  staticListRequestSet

Description

This function creates a dataframe set of static list requests for a constituent list
Usage

```
staticListRequestSet(
    mydsws = dsws$new(),
    instrument,
    expression = "",
    endDate = Sys.Date(),
    frequency = "D"
)
```

Arguments

- `mydsws`: a dsws object, if not provided a new one will be created
- `instrument`: array of instruments
- `expression`: an array of expressions such as PCH#(XXXX,1M)
- `endDate`: the date of the request
- `frequency`: optional frequency defaults to "D"

Details

This function creates a dataframe set of static list requests for a constituent list

Value

- a dataframe of the data

________

staticRequest  
make a static request (Depreciated)

Description

makes a static (or snapshot request) from the Datastream DSWS server

Usage

```
staticRequest(
    dwei = getDataStream(),
    DSCode,
    Expression = "",
    endDate = Sys.Date(),
    frequency = "D",
    verbose = FALSE,
    noCache = FALSE
)
```
**Arguments**

dwei - A Datastream Client Interface object created with getDataStream
DSCode - an array of instruments eg c("RIO","MKS")
Expression - the data to return eg MNEM or NAME
endDate - the date of the request, or the string "TODAY"
frequency - the frequency of the request
verbose - whether to give messages during the request
noCache - no longer used

**Details**

staticRequest Function that returns a the value of Expression for the array of instruments in DSCode from Datastream parameters are

**Value**

returns an array of the requested information

**Description**

This function creates a dataframe set of static requests for a set of stocks/indices

**Usage**

```
staticRequestSet(
  mydsws = dsws$new(),
  instrument,
  expression = "",
  endDate = Sys.Date(),
  frequency = "D",
  verbose = FALSE
)
```

**Arguments**

mydsws - a dsws object, if not provided a new one will be created
instrument - array of instruments
expression - an array of expressions such as PCH#(XXXX,1M) or Dataitems
endDate - the date of the request
frequency - optional frequency defaults to "D"
verbose - whether to display messages as making the request
**timeSeriesListRequest**

**Details**

return a dataframe of static data

**Value**

a dataframe of the data

---

**timeSeriesListRequest**  
make a `timeSeries` request for a list *(Depreciated)*

**Description**

make a timeseries request for a constituent list from Datastream DSWS  
**timeSeriesListRequest**  
Function that returns a timeseries from Datastream constituent list parameters are

**Usage**

```r

timeSeriesListRequest(
  dwei = getDataStream(),
  DSCode,
  Instrument,
  startDate,
  endDate = Sys.Date(),
  frequency = "D",
  sStockList,
  aTimeSeries,
  verbose = FALSE
)
```

**Arguments**

- **dwei** - A Datastream Client Interface object created with `getDataStream`
- **DSCode** - the constituent list requested eg 'LFTSE100'
- **Instrument** - the expression to return for each member of constituent list
- **startDate** - the start date of the timeseries
- **endDate** - the end date of the timeseries
- **frequency** - the frequency of the request
- **sStockList** - variable that is returned with list of of the stocks
- **aTimeSeries** - variable that is returned with the set of timeseries
- **verbose** - whether to give messages during the request

**Details**

List request
Value

whether the request has been successful, but also in sStockList: a list a two element vector of the
displayname and symbol for each timeseries in aTimeseries: a list of class xts with the requested
timeseries information

timeSeriesRequest  make a timeseries request (Depreciated)

Description

make a timeseries request from the Datastream DSWS server

Usage

timeSeriesRequest(
  dwei = getDataStream(),
  DSCodes = "",
  Instrument = "",
  startDate = Sys.Date(),
  endDate = Sys.Date(),
  frequency = "D",
  sStockList,
  aTimeSeries,
  myType = "numeric",
  verbose = FALSE
)

Arguments

dwei - A Datastream Client Interface object created with getDataStream
DSCodes - one or more codes to return, eg "MKS" or c("MKS","SAB")
Instrument - the instrument or expression to return eg PCH#(XXXX,1M)
startDate - the start date of the timeseries
endDate - the end date of the timeseries
frequency - the frequency of the request
sStockList - variable that is returned with list of the stocks
aTimeSeries - variable that is returned with the set of timeseries. This is a list that is not
guaranteed to be in the same order as sStockList
myType - the type of the return values eg numeric (default), Date or Character
verbose - whether to give messages during the request

Details

function timeSeriesRequest obtains a timeseries from Datastream
**Value**

whether the request has been successful in sStockList: a list a two element vector of the displayname and symbol for each timeseries in aTimeseries: a list of class xts with the requested timeseries information

---

**Description**

Uploads and appends an xts into a UCTS in the Datastream Database

**Usage**

```r
UCTSAppend(  
  tsData,  
  TSCode = "",  
  MGMTGroup = "ABC",  
  freq = c("D", "W", "M", "Q", "Y"),  
  seriesName,  
  Units = "",  
  Decimals = 2,  
  ActPer = c("N", "Y"),  
  freqConversion = c("ACT", "SUM", "AVG", "END"),  
  Alignment = c("1ST", "MID", "END"),  
  Carry = c("YES", "NO", "PAD"),  
  PrimeCurr = "",  
  overwrite = TRUE,  
  strUsername = ifelse(Sys.getenv("DatastreamUsername") != ",",  
    Sys.getenv("DatastreamUsername"), options()$Datastream.Username),  
  strPassword = ifelse(Sys.getenv("DatastreamPassword") != ",",  
    Sys.getenv("DatastreamPassword"), options()$Datastream.Password),  
  strServerName = "http://product.datastream.com",  
  strServerPage = "/UCTS/UCTSMaint.asp"  
)
```

**Arguments**

- `tsData` - an xts (or timeseries object that can be converted to one) to be uploaded.
- `TSCode` - The mnemonic of the target UCTS
- `MGMTGroup` - Must have management group. Only the first characters will be used.
- `freq` - The frequency of the data to be uploaded
- `seriesName` - the name of the series
- `Units` - Units of the data - can be no more than 12 characters - excess will be trimmed to that length
Decimals: Number of Decimals in the data - a number between 0 and 9 - if outside that range then trimmed

ActPer: Whether the values are percentages ("N") or actual numbers ("Y")

freqConversion: How to do any FX conversions

Alignment: Alignment of the data within periods

Carry: Whether to carry data over missing dates

PrimeCurr: The currency of the timeseries

overwrite: if TRUE then existing data in the UCTS will be overwritten

strUsername: Your Datastream username

strPassword: Your Datastream Password

strServerName: URL of the Datastream server

strServerPage: Page on the Datastream server

Details

This function checks if there is a pre-existing timeseries already in Datastream. If there is then it will append the xts onto the existing series. If there are any overlapping dates then depending on the setting of overwrite then the new data will overwrite the existing data in the UCTS.

Value

TRUE if the upload has been a success, otherwise an error message

Description

Uploads an xts into a UCTS in the Datastream Database

Usage

```r
UCTSUpload(
  tsData,
  TSCode = "",
  MGMTGroup = "ABC",
  freq = c("D", "W", "M", "Q", "Y"),
  seriesName,
  Units = "",
  Decimals = 2,
  ActPer = c("N", "Y"),
  freqConversion = c("ACT", "SUM", "AVG", "END"),
  Alignment = c("1ST", "MID", "END"),
  Carry = c("YES", "NO", "PAD"),
)```

}

Arguments

tsData - an xts (or timeseries object that can be converted to one) to be uploaded.
TSCode The mnemonic of the target UCTS
MGMTGroup Must have management group. Only the first characters will be used.
freq The frequency of the data to be uploaded
seriesName the name of the series
Units Units of the data - can be no more than 12 characters - excess will be trimmed to that length
Decimals Number of Decimals in the data - a number between 0 and 9 - if outside that range then trimmed
ActPer Whether the values are percentages ("N") or actual numbers ("Y")
freqConversion How to do any FX conversions
Alignment Alignment of the data within periods
Carry whether to carry data over missing dates
PrimeCurr the currency of the timeseries
strUsername your Datastream username
strPassword your Datastream Password
strServerName URL of the Datastream server
strServerPage page on the datastream server

Details

Note this function does not check to see if there is a pre-existing timeseries already in Datastream. It will just overwrite any existing UCTS.

Value

TRUE if the upload has been a success, otherwise an error message
Index

* datasets
  currencyDS2ISO, 7
  cbindRobust, 2
classconstructor, 2
currencyDS2ISO, 7

DataStreamDSWS2R, 8
dsws/classconstructor, 2
dataStream, 8
listRequest, 9
myStaticRequestSet, 10
staticListRequestSet, 10
staticRequest, 11
staticRequestSet, 12

timeSeriesListRequest, 13
timeSeriesRequest, 14

UCTSAppend, 15
UCTSUpload, 16