

Package ‘RJSDMX’

March 9, 2017

Version 1.7

Title R Interface to SDMX Web Services

Author Attilio Mattiocco, Diana Nicoletti, Gianpaolo Lopez, Banca d'Italia

Maintainer Attilio Mattiocco <attilio.mattiocco@bancaditalia.it>

Description Provides functions to retrieve data and metadata from providers that disseminate data by means of SDMX web services.

SDMX (Statistical Data and Metadata eXchange) is a standard that has been developed with the aim of simplifying the exchange of statistical information.

More about the SDMX standard and the SDMX Web Services can be found at: <http://sdmx.org> .

Depends R (>= 2.12.0), rJava (>= 0.8-8), zoo (>= 1.6-4)

SystemRequirements Java (>= 1.6)

License EUPL

URL <https://github.com/amattioc/SDMX/>

BugReports <https://github.com/amattioc/SDMX/issues>

Copyright Banca d'Italia

NeedsCompilation no

Repository CRAN

Date/Publication 2017-03-09 18:08:34

R topics documented:

RJSDMX-package	2
addProvider	3
getCodes	4
getDimensions	4
getDSDIdentifier	5
getFlows	6
getProviders	6
getSDMX	7

getTimeSeries	8
getTimeSeriesRevisions	8
getTimeSeriesTable	9
sdmxdf	10
sdmxHelp	11

Index	12
--------------	-----------

RJSMDX-package	<i>Gets timeseries from SDMX data Provider</i>
----------------	--

Description

This package provides functions to extract timeseries data and structural metadata from an SDMX Provider (e.g. ECB, OECD, EUROSTAT) via SDMX Web Service

Details

Package: RJSMDX
Type: Package

The SDMX Connectors framework (of which RJSMDX is part) aims to offer data users the means for efficiently interacting with SDMX Web Service providers from within the most popular statistical tools. The source code of the SDMX Connectors project can be found at:

<https://github.com/amattioc/SDMX>

Information about the R Connector can be found in the dedicated wiki page:

<https://github.com/amattioc/SDMX/wiki/RJSMDX:-Connector-for-R>

In particular, all information related to configuration (network, tracing, security) can be found at:

<https://github.com/amattioc/SDMX/wiki/Configuration>

Author(s)

Attilio Mattiocco, Diana Nicoletti, Bank of Italy, IT Support for the Economic Research <attilio.mattiocco@bancaditalia.it>
diana.nicoletti@bancaditalia.it

References

<http://sdmx.org/>

See Also

getProviders, getTimeSeries, sdmxHelp

Examples

```
## Not run:
my_ts = getTimeSeries('ECB', 'EXR.M.USD.EUR.SP00.A')

## End(Not run)
```

addProvider	<i>add new provider</i>
-------------	-------------------------

Description

Configure a new data provider (only SDMX 2.1 REST providers are supported). This function can be used to configure a new (SDMX 2.1 compliant, REST based) data provider.

Usage

```
addProvider(name, endpoint, needsCredentials = FALSE, needsURLencoding = FALSE,
  supportsCompression = TRUE, description = "")
```

Arguments

name	the name of the provider
endpoint	the URL where the provider resides
needsCredentials	set this to TRUE if the user needs to authenticate to query the provider
needsURLencoding	set this to TRUE if the provider does not handle character '+' in URLs
supportsCompression	set this to TRUE if the provider is able to handle compression
description	a brief text description of the provider

Examples

```
## Not run:
addProvider('test', 'http://sdw-wsrest.ecb.europa.eu/service', FALSE)
getProviders()

## End(Not run)
```

getCodes *get dsd codes for dataflow*

Description

Extract the codes of a dimension. This function is used to retrieve the list of codes available for the input dimension and flow.

Usage

```
getCodes(provider, flow, dimension)
```

Arguments

flow	the identifier of the dataflow
dimension	the identifier of the dimension
provider	the name of the provider

Details

```
getCodes(provider, dataflow, dimension)
```

Examples

```
## Not run:  
codes=getCodes('ECB', 'EXR', 'FREQ')  
  
## End(Not run)
```

getDimensions *get dsd dimensions for dataflow*

Description

Extract the dimensions of a DataFlow. This function is used to retrieve the list of dimensions of the input dataflow

Usage

```
getDimensions(provider, dataflow)
```

Arguments

dataflow	the identifier of the dataflow
provider	the name of the provider

Details

```
getDimensions(provider, dataflow)
```

Examples

```
## Not run:  
dims = getDimensions('ECB', 'EXR')  
  
## End(Not run)
```

<code>getDSDIdentifier</code>	<i>get DSD Identifier for dataflow</i>
-------------------------------	--

Description

Extract the dsd identifier of a DataFlow. This function is used to retrieve the name of the keyfamily of the input dataflow.

Usage

```
getDSDIdentifier(provider, dataflow)
```

Arguments

<code>provider</code>	the name of the provider
<code>dataflow</code>	the identifier of the dataflow

Details

```
getDSDIdentifier(provider, dataflow)
```

Examples

```
## Not run:  
id = getDSDIdentifier('ECB', 'EXR')  
  
## End(Not run)
```

getFlows	<i>get provider flow list</i>
----------	-------------------------------

Description

Extract the list of DataFlows of a provider. This function is used to query the list of dataflows of the provider. A matching pattern can be provided, if needed.

Usage

```
getFlows(provider, pattern = "")
```

Arguments

pattern	the pattern to match against the dataflow id or description. If a pattern is not provided, all dataflows are returned.
provider	the name of the provider

Details

```
getFlows(provider, pattern)
```

Author(s)

Attilio Mattiocco <Attilio.Mattiocco@bancaditalia.it>, Diana Nicoletti

Examples

```
## Not run:
## get all flows from ECB
flows = getFlows('ECB')
## get all flows that contain the 'EXR'
flows = getFlows('ECB', '*EXR*')

## End(Not run)
```

getProviders	<i>get available providers</i>
--------------	--------------------------------

Description

Extract the list of available Data Providers. This function is used to query the list of data providers.

Usage

```
getProviders()
```

Details

```
getProviders()
```

Examples

```
## Not run:
getProviders()

## End(Not run)
```

getSDMX	<i>get data</i>
---------	-----------------

Description

Extract a list of time series. This function is exactly the same as `getTimeSeries`.

Usage

```
getSDMX(provider, id, start = "", end = "")
```

Arguments

<code>id</code>	identifier of the time series
<code>provider</code>	the name of the provider
<code>end</code>	the end time - optional
<code>start</code>	the start time - optional

Details

```
getSDMX(provider, dataflow, start, end)
```

Examples

```
## Not run:
## get single time series: EXR.A.USD.EUR.SP00.A (or: EXR/A+M.USD.EUR.SP00.A)
my_ts=getSDMX('ECB', 'EXR.A.USD.EUR.SP00.A')
## get monthly and annual frequency: 'EXR.A|M.USD.EUR.SP00.A' (or: EXR/A+M.USD.EUR.SP00.A)
my_ts=getSDMX('ECB', 'EXR.A|M.USD.EUR.SP00.A')
## get all available frequencies: 'EXR.*.USD.EUR.SP00.A' (or: EXR/.USD.EUR.SP00.A)
my_ts=getSDMX('ECB', 'EXR.*.USD.EUR.SP00.A')

## End(Not run)
```

getTimeSeries	<i>get time series</i>
---------------	------------------------

Description

Extract a list of time series. This function is used to extract a list of time series identified by the parameters provided in input. `getTimeSeries(provider, dataflow, start, end)`

Usage

```
getTimeSeries(provider, id, start = "", end = "")
```

Arguments

id	identifier of the time series
provider	the name of the provider
end	the end time - optional
start	the start time - optional

Examples

```
## Not run:
## get single time series: EXR.A.USD.EUR.SP00.A (or: EXR/A+M.USD.EUR.SP00.A)
my_ts=getTimeSeries('ECB', 'EXR.A.USD.EUR.SP00.A')
## get monthly and annual frequency: 'EXR.A|M.USD.EUR.SP00.A' (or: EXR/A+M.USD.EUR.SP00.A)
my_ts=getTimeSeries('ECB', 'EXR.A|M.USD.EUR.SP00.A')
## get all available frequencies: 'EXR.*.USD.EUR.SP00.A' (or: EXR/.USD.EUR.SP00.A)
my_ts=getTimeSeries('ECB', 'EXR.*.USD.EUR.SP00.A')

## End(Not run)
```

getTimeSeriesRevisions	<i>get data revisions</i>
------------------------	---------------------------

Description

Extract a list of time series starting from a specific update time and with history of revisions. This function works as **getTimeSeries** and **getSDMX** but the query can be narrowed to getting only observations that were updated after a specific point in time, and eventually it returns the revision history of the matching time series. This means that the result list can contain multiple instances of the same time series, each with the specific action and validity attributes.

Usage

```
getTimeSeriesRevisions(provider, id, start = "", end = "",
  updatedAfter = "", includeHistory = TRUE)
```

Arguments

id	identifier of the time series
provider	the name of the provider
end	the end time - optional
start	the start time - optional
updatedAfter	the updatedAfter time - optional. It has to be in the form: 'YYYY-MM-DD'
includeHistory	boolean parameter - optional. If TRUE the full list of revisions will be returned

Details

```
getTimeSeriesRevisions(provider, id, start, end, updatedAfter, includeHistory)
```

Examples

```
## Not run:
# get single time series with history:
my_ts=getTimeSeriesRevisions('ECB', 'EXR.A.USD.EUR.SP00.A', includeHistory=TRUE)

# get single time series (only observations updated after january 1st 2015):
my_ts=getTimeSeriesRevisions('ECB', 'EXR.A.USD.EUR.SP00.A',
  updatedAfter='2015', includeHistory=FALSE)

# get single time series (full revision history starting from january 1st 2015):
my_ts=getTimeSeriesRevisions('ECB', 'EXR.A.USD.EUR.SP00.A',
  updatedAfter='2015', includeHistory=TRUE)

## End(Not run)
```

```
getTimeSeriesTable    get time series and return a data.frame
```

Description

Extract a list of time series identified by the parameters provided in input, and return a data.frame as result. getTimeSeriesTable(provider, dataflow, start, end)

Usage

```
getTimeSeriesTable(provider, id, start = "", end = "")
```

Arguments

id	identifier of the time series
provider	the name of the provider
end	the end time - optional
start	the start time - optional

Examples

```
## Not run:
# get single time series: EXR.A.USD.EUR.SP00.A (alternatively: EXR/A+M.USD.EUR.SP00.A)
my_df=getTimeSeriesTable('ECB', 'EXR.A.USD.EUR.SP00.A')

# get monthly and annual frequency: 'EXR.A|M.USD.EUR.SP00.A'
# (alternatively: EXR/A+M.USD.EUR.SP00.A)
my_df=getTimeSeriesTable('ECB', 'EXR.A|M.USD.EUR.SP00.A')

# get all available frequencies: 'EXR.*.USD.EUR.SP00.A'
# (alternatively: EXR/.USD.EUR.SP00.A)
my_df=getTimeSeriesTable('ECB', 'EXR.*.USD.EUR.SP00.A')

## End(Not run)
```

sdmxdf

convert time series to data.frame

Description

This function is used to transform the output of the `getSDMX` (or `getTimeSeries`) functions from a list of time series to a `data.frame`. The metadata can be requested by explicitly passing the appropriate parameters.

Usage

```
sdmxdf(tslist, meta = FALSE, id = TRUE)
```

Arguments

tslist	the list of time series to be converted
meta	set this to TRUE if you want metadata to be included (default: FALSE, as this may increase the size of the result quite a bit)
id	set this to FALSE if you do not want the time series id to be included (default: TRUE)

Details

```
sdmxdf()
```

Examples

```
## Not run:
a=getSDMX('ECB', 'EXR.A|Q|M|D.USD.EUR.SP00.A')
ddf = sdmxdf(a)
ddf = sdmxdf(a, meta=TRUE)

## End(Not run)
```

`sdmxHelp`*open helper*

Description

Open a helper graphical application. This function opens a small sdmx metadata browser that can be helpful when building queries.

Usage

```
sdmxHelp(internalJVM = T)
```

Arguments

`internalJVM` TRUE (default) if the GUI has to live in the R JVM. Set this to FALSE in MAC, to avoid issue #41

Details

```
sdmxHelp()
```

Examples

```
## Not run:
#opens the helper in the R JVM
sdmxHelp()
#opens the helper in an external JVM
sdmxHelp(FALSE)

## End(Not run)
```

Index

*Topic **package**

RJSDMX-package, [2](#)

*Topic **rJava**

getFlows, [6](#)

addProvider, [3](#)

getCodes, [4](#)

getDimensions, [4](#)

getDSDIdentifier, [5](#)

getFlows, [6](#)

getProviders, [6](#)

getSDMX, [7](#)

getTimeSeries, [8](#)

getTimeSeriesRevisions, [8](#)

getTimeSeriesTable, [9](#)

RJSDMX (RJSDMX-package), [2](#)

RJSDMX-package, [2](#)

sdmxdf, [10](#)

sdmxHelp, [11](#)