Package ‘rdbnomics’

October 14, 2022

Type Package

Title Download DBnomics Data

Version 0.6.4

Description R access to hundreds of millions data series from DBnomics API (<https://db.nomics.world/>).

Depends R (>= 3.1.0)

License AGPL-3

URL https://git.nomics.world/dbnomics/rdbnomics

BugReports https://git.nomics.world/dbnomics/rdbnomics/-/issues

Encoding UTF-8

LazyData true

Imports curl, data.table, jsonlite

Suggests knitr, rmarkdown, tinytest

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Author Sebastien Galais [cre, ctb], Thomas Brand [aut]

Maintainer Sebastien Galais <s915.stem@gmail.com>

Repository CRAN

Date/Publication 2020-10-25 20:10:02 UTC

R topics documented:

dbnomics ....................................................... 2
rdb .......................................................... 3
rdbnomics ..................................................... 7
rdb_by_api_link ............................................ 7
rdb_datasets ................................................. 10
Description

dbnomics is a simple ggplot2 theme for drawing nicer graphics. We do not recommend to use it. It has been included in the package to avoid errors when reproducing the vignette examples.

Usage

dbnomics(color_palette = "Set1", ...)

Arguments

color_palette Character string (default "Set1") to change the default color palette. If you want to use the default palette, set it to NULL.

... Arguments to be passed to the function ggplot2::theme.

Author(s)

Sebastien Galais

Examples

## Not run:
library(magrittr)
library(ggplot2)

rdb("IMF", "WEO:2019-10", query = "France current account balance percent") %>%
ggplot(aes(x = period, y = value, color = series_name)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  dbnomics()

## End(Not run)
Description

`rdb` downloads data series from DBnomics using shortcuts like `ids`, `dimensions`, `mask`, `query` or using an `api_link`.

Usage

```r
rdb(
    provider_code = NULL,
    dataset_code = NULL,
    ids = NULL,
    dimensions = NULL,
    mask = NULL,
    query = NULL,
    api_link = NULL,
    filters = getOption("rdbnomics.filters"),
    use_readLines = getOption("rdbnomics.use_readLines"),
    curl_config = getOption("rdbnomics.curl_config"),
    verbose = getOption("rdbnomics.verbose_warning"),
    ...
)
```

Arguments

- `provider_code` Character string (default NULL). DBnomics code of the provider.
- `dataset_code` Character string (default NULL). DBnomics code of the dataset.
- `ids` Character string (default NULL). DBnomics code of one or several series.
- `dimensions` List or character string (single quoted) (default NULL). DBnomics code of one or several dimensions in the specified provider and dataset. If it is a named list, then the function `toJSON` (from the package `jsonlite`) is applied to generate the JSON object.
- `mask` Character string (default NULL). DBnomics code of one or several masks in the specified provider and dataset.
- `query` Character string (default NULL). A query to filter/select series from a provider’s dataset.
- `api_link` Character string. DBnomics API link of the search. It should starts with `http://` or `https://`.
- `filters` List (default NULL). This argument must be a named list for one filter because the function `toJSON` of the package `jsonlite` is used before sending the request to the server. For multiple filters, you have to provide a list of valid filters (see examples). A valid filter is a named list with an element `code` which is a character string,
and an element parameters which is a named list with elements frequency and method or a NULL.

use_readLines Logical (default FALSE). If TRUE, then the data are requested and read with the base function readLines i.e. through the default R internet connection. This can be used to get round the error Could not resolve host: api.db.nomics.world.

curl_config Named list (default NULL). If not NULL, it is used to configure a proxy connection. This configuration is passed to the function curl_fetch_memory of the package curl. A temporary curl_handle object is created internally with arguments equal to the provided list in curl_config. For curl_fetch_memory arguments see curl_fetch. For available curl options see curl_options, names(curl_options()) and libcurl.

verbose Logical (default FALSE). Show warnings of the function.

... Arguments to be passed to the internal function .rdb.

Details

This function gives you access to hundreds of millions data series from DBnomics API (documentation about the API can be found here). The code of each series is given on the DBnomics website.

In the event that only the argument ids is provided (and those in the ellipsis ...), the argument name can be dropped. The character string vector is directly passed to ids.

If only the argument api_link is provided (and those in the ellipsis ...), then the argument name can be dropped. The character string vector is directly passed to api_link.

In the same way, if only provider_code, dataset_code and mask are provided then the arguments names can be dropped. The last character string is automatically passed to mask.

Value

A data.table.

Author(s)

Sebastien Galais

Examples

## Not run:
## By ids
# Fetch one series from dataset 'Unemployment rate' (ZUTN) of AMECO provider:
df1 <- rdb(ids = "AMECO/ZUTN/EA19.1.0.0.0.ZUTN")
# or when no argument names are given (provider_code -> ids)
# df1 <- rdb("AMECO/ZUTN/EA19.1.0.0.0.ZUTN")

# Fetch two series from dataset 'Unemployment rate' (ZUTN) of AMECO provider:
df2 <- rdb(ids = c("AMECO/ZUTN/EA19.1.0.0.0.ZUTN", "AMECO/ZUTN/DNK.1.0.0.0.ZUTN"))

# Fetch two series from different datasets of different providers:
df3 <- rdb(ids = c("AMECO/ZUTN/EA19.1.0.0.0.ZUTN", "IMF/BOP/A.FR.BCA_BP6_EUR"))
## By dimensions

### Fetch one value of one dimension from dataset 'Unemployment rate' (ZUTN) of AMECO provider:

```r
df1 <- rdb("AMECO", "ZUTN", dimensions = list(geo = "ea12"))
```

### Fetch two values of one dimension from dataset 'Unemployment rate' (ZUTN) of AMECO provider:

```r
df2 <- rdb("AMECO", "ZUTN", dimensions = list(geo = c("ea12", "dnk"))
```

### Fetch several values of several dimensions from dataset 'Doing business' (DB) of World Bank:

```r
dim <- list(
  country = c("DZ", "PE"),
  indicator = c("ENF.CONT.COEN.COST.ZS", "IC.REG.COST.PC.FE.ZS")
)
df3 <- rdb("WB", "DB", dimensions = dim)
```

## By mask

### Fetch one series from dataset 'Balance of Payments' (BOP) of IMF:

```r
df1 <- rdb("IMF", "BOP", mask = "A.FR.BCA_BP6_EUR")
```

### Fetch two series from dataset 'Balance of Payments' (BOP) of IMF:

```r
df2 <- rdb("IMF", "BOP", mask = "A.FR+ES.BCA_BP6_EUR")
```

### Fetch all series along one dimension from dataset 'Balance of Payments' (BOP) of IMF:

```r
df3 <- rdb("IMF", "BOP", mask = "A..BCA_BP6_EUR")
```

### Fetch series along multiple dimensions from dataset 'Balance of Payments' (BOP) of IMF:

```r
df4 <- rdb("IMF", "BOP", mask = "A.FR.BCA_BP6_EUR+IA_BP6_EUR")
```

## By query

### Fetch one series from dataset 'WEO by countries (2019-10 release)' (WEO:2019-10) from IMF:

```r
df1 <- rdb("IMF", "WEO:2019-10", query = "France current account balance percent")
```

### Fetch series from dataset 'WEO by countries (2019-10 release)' (WEO:2019-10) from IMF:

```r
df2 <- rdb("IMF", "WEO:2019-10", query = "current account balance percent")
```

## By api_link

### Fetch two series from different datasets of different providers:

```r
df1 <- rdb(
  api_link = paste0("https://api.ameco.org/rdb/v2/", 
                    "provider/AMECO", 
                    ",dataset/ZUTN", 
                    ",dimension/geo", 
                    ",value/ea12")
```

### Fetch two series from different datasets of different providers:

```r
df2 <- rdb(
  api_link = paste0("https://api.ameco.org/rdb/v2/", 
                    "provider/AMECO", 
                    ",dataset/ZUTN", 
                    ",dimension/geo", 
                    "=ea12,dnk")
```

### Fetch several values of several dimensions from dataset 'Doing business' (DB) of World Bank:

```r
dim <- list(
  country = c("DZ", "PE"),
  indicator = c("ENF.CONT.COEN.COST.ZS", "IC.REG.COST.PC.FE.ZS")
)
df3 <- rdb("WB", "DB", dimensions = dim)
```

### Fetch series along multiple dimensions from dataset 'Balance of Payments' (BOP) of IMF:

```r
df4 <- rdb("IMF", "BOP", mask = "A.FR.BCA_BP6_EUR+IA_BP6_EUR")
```
"https://api.db.nomics.world/v22/",
"series?observations=1&series_ids=AMECO/ZUTN/EA19.1.0.0.0.ZUTN,IMF/CPI/A.AT.PCPIT_IX"
)
)

# Fetch one series from the dataset 'Doing Business' of WB provider:
df2 <- rdb(
  api_link = paste0(
    "https://api.db.nomics.world/v22/series/WB/DB?dimensions=%7B%22
    "indicator%22%3A%5B%22IC.REG.PROC.FE.NO%22%5D%7D&q=Doing%20Business",
    "&observations=1&format=json&align_periods=1&offset=0&facets=0"
  )
)
# or when no argument names are given (provider_code -> api_link)
df1 <- rdb(
  paste0(
    "https://api.db.nomics.world/v22/",
    "series?observations=1&series_ids=AMECO/ZUTN/EA19.1.0.0.0.ZUTN,IMF/CPI/A.AT.PCPIT_IX"
  )
)

## Use a specific proxy to fetch the data
# Fetch one series from dataset 'Unemployment rate' (ZUTN) of AMECO provider:
h <- list(
  proxy = "<proxy>",
  proxyport = <port>,
  proxyusername = "<username>",
  proxypassword = "<password>"
)
options(rdbnomics.curl_config = h)
df1 <- rdb(ids = "AMECO/ZUTN/EA19.1.0.0.0.ZUTN")
# or to use once
options(rdbnomics.curl_config = NULL)
df1 <- rdb(ids = "AMECO/ZUTN/EA19.1.0.0.0.ZUTN", curl_config = h)

## Use R default connection to avoid a proxy failure (in some cases)
# Fetch one series from dataset 'Unemployment rate' (ZUTN) of AMECO provider:
options(rdbnomics.use_readLines = TRUE)
df1 <- rdb(ids = "AMECO/ZUTN/EA19.1.0.0.0.ZUTN")
# or to use once
df1 <- rdb(ids = "AMECO/ZUTN/EA19.1.0.0.0.ZUTN", use_readLines = TRUE)

## Apply filter(s) to the series
# One filter
df1 <- rdb(
  filters = list(
    code = "interpolate",
    parameters = list(frequency = "daily", method = "spline")
  )
)
```r
## Two filters
df1 <- rdb(
  filters = list(
    list(
      code = "interpolate",
      parameters = list(frequency = "quarterly", method = "spline")
    ),
    list(
      code = "aggregate",
      parameters = list(frequency = "annual", method = "average")
    )
  )
)

## End(Not run)
```

**rdbnomics**

Package *rdbnomics*

**Description**

DBnomics R client (<https://db.nomics.world/>).

---

**rdb_by_api_link**

Download DBnomics data using API link (deprecated).

**Description**

*rdb_by_api_link* downloads data series from DBnomics.

**Usage**

```r
rdb_by_api_link(
  api_link, 
  use_readLines = getOption("rdbnomics.use_readLines"),
  curl_config = getOption("rdbnomics.curl_config"),
  filters = getOption("rdbnomics.filters")
)
```
### Arguments

- **api_link**  
  Character string. DBnomics API link of the search.

- **use_readLines**  
  Logical (default FALSE). If TRUE, then the data are requested and read with the base function `readLines` i.e. through the default R internet connection. This can be used to get round the error `Could not resolve host: api.db.nomics.world`.

- **curl_config**  
  Named list (default NULL). If not NULL, it is used to configure a proxy connection. This configuration is passed to the function `curl_fetch_memory` of the package `curl`. A temporary `curl_handle` object is created internally with arguments equal to the provided list in `curl_config`. For `curl_fetch_memory` arguments see `curl_fetch`. For available curl options see `curl_options`, `names(curl_options())` and `libcurl`.

- **filters**  
  List (default NULL). This argument must be a named list for one filter because the function `toJSON` of the package `jsonlite` is used before sending the request to the server. For multiple filters, you have to provide a list of valid filters (see examples). A valid filter is a named list with an element `code` which is a character string, and an element `parameters` which is a named list with elements frequency and method or a NULL.

### Details

This function gives you access to hundreds of millions data series from DBnomics API (documentation about the API can be found [here](https://www.db-nomics.com)). The API link is given on the DBnomics website.

### Value

A `data.table`.

### Author(s)

Sebastien Galais

### See Also

- `rdb`

### Examples

```r
## Not run:
# Fetch two series from different datasets of different providers:
df1 <- rdb_by_api_link(
  paste0(
    "https://api.db.nomics.world/v22/",
    "series?observations=1&series_ids=AMECO/ZUTN/EA19.1.0.0.0.ZUTN,IMF/CPI/A.AT.PCPIT_IX"
  )
)

# Fetch one series from the dataset 'Doing Business' of WB provider:
df2 <- rdb_by_api_link(
```

rdb_by_api_link

paste0(
  "https://api.db.nomics.world/v22/series/WB/DB?dimensions=%7B%22",
  "indicator%22%3A%5B%22IC.REG.PROC.FE.NO%22%5D&q=Doing%20Business",
  "&observations=1&format=json&align_periods=1&offset=0&facets=0"
)
)

## Use a specific proxy to fetch the data
# Fetch one series from the dataset 'Doing Business' of WB provider:

h <- list(
  proxy = "<proxy>",
  proxyport = <port>,
  proxyusername = "<username>",
  proxypassword = "<password>"
)

options(rdbnomics.curl_config = h)
df2 <- rdb_by_api_link(
  paste0(
    "https://api.db.nomics.world/v22/series/WB/DB?dimensions=%7B%22",
    "indicator%22%3A%5B%22IC.REG.PROC.FE.NO%22%5D&q=Doing%20Business",
    "&observations=1&format=json&align_periods=1&offset=0&facets=0"
  ),
  curl_config = h
)

# or to use once
df2 <- rdb_by_api_link(
  paste0(
    "https://api.db.nomics.world/v22/series/WB/DB?dimensions=%7B%22",
    "indicator%22%3A%5B%22IC.REG.PROC.FE.NO%22%5D&q=Doing%20Business",
    "&observations=1&format=json&align_periods=1&offset=0&facets=0"
  ),
  use_readLines = TRUE
)

## Use R default connection to avoid a proxy failure (in some cases)
# Fetch one series from the dataset 'Doing Business' of WB provider:

options(rdbnomics.use_readLines = TRUE)
df2 <- rdb_by_api_link(
  paste0(
    "https://api.db.nomics.world/v22/series/WB/DB?dimensions=%7B%22",
    "indicator%22%3A%5B%22IC.REG.PROC.FE.NO%22%5D&q=Doing%20Business",
    "&observations=1&format=json&align_periods=1&offset=0&facets=0"
  ),
  use_readLines = TRUE
)
rdb_databases

## Apply filter(s) to the series

# One filter

def3 <- rdb_by_api_link(
  filters = list(
    code = "interpolate",
    parameters = list(frequency = "daily", method = "spline")
  )
)

# Two filters

def3 <- rdb_by_api_link(
  filters = list(
    list(
      code = "interpolate",
      parameters = list(frequency = "quarterly", method = "spline")
    ),
    list(
      code = "aggregate",
      parameters = list(frequency = "annual", method = "average")
    )
  )
)

## End(Not run)

rdb_databases  Download list of datasets for DBnomics providers.

Description

rdb_databases downloads the list of available datasets for a selection of providers (or all of them) from DBnomics.

Usage

rdb_databases(
  provider_code = NULL,
  use_readLines =getOption("rdbnomics.use_readLines"),
  curl_config =getOption("rdbnomics.curl_config"),
  simplify = FALSE,
  ...
)
Arguments

- **provider_code** | Character string (default NULL). DBnomics code of one or multiple providers. If NULL, the providers are firstly downloaded with the function `rdb_providers` and then the available datasets are requested.

- **use_readLines** | Logical (default FALSE). If TRUE, then the data are requested and read with the base function `readLines` i.e. through the default R internet connection. This can be used to get round the error `Could not resolve host: api.db.nomics.world`.

- **curl_config** | Named list (default NULL). If not NULL, it is used to configure a proxy connection. This configuration is passed to the function `curl_fetch_memory` of the package `curl`. A temporary curl_handle object is created internally with arguments equal to the provided list in `curl_config`. For `curl_fetch_memory` arguments see `curl_fetch`. For available curl options see `curl_options`, `names(curl_options())` and `libcurl`.

- **simplify** | Logical (default FALSE). If TRUE, when the datasets are requested for only one provider then a `data.table` is returned, not a list of `data.tables`.

- **...** | Additional arguments.

Details

By default, the function returns a named list of `data.tables` containing the datasets of the providers from DBnomics.

Value

A named list of `data.tables` or a `data.table`.

Author(s)

Sebastien Galais

See Also

- `rdb_providers`, `rdb_last_updates`, `rdb_dimensions`, `rdb_series`

Examples

```r
## Not run:
rdb_datasets(provider_code = "IMF")

rdb_datasets(provider_code = "IMF", simplify = TRUE)

rdb_datasets(provider_code = c("IMF", "BDF"))

options(rdbnomics.progress_bar_datasets = TRUE)
rdb_datasets()
options(rdbnomics.progress_bar_datasets = FALSE)

rdb_datasets(provider_code = "IMF", use_readLines = TRUE)
```
rdb_dimensions

rdb_dimensions(
  provider_code = "IMF",
  curl_config = list(proxy = "<proxy>", proxyport = <port>)
)

## End(Not run)

rdb_dimensions  Download list of dimensions for datasets of DBnomics providers.

Description

rdb_dimensions downloads the list of dimensions (if they exist) for available datasets of a selection of providers from DBnomics.

Usage

rdb_dimensions(
  provider_code = NULL,
  dataset_code = NULL,
  use_readLines = getOption("rdbnomics.use_readLines"),
  curl_config = getOption("rdbnomics.curl_config"),
  simplify = FALSE,
  ...
)

Arguments

Provider_code  Character string (default NULL). DBnomics code of one or multiple providers. If NULL, the providers are firstly downloaded with the function rdb_providers and then the datasets are requested.

dataset_code  Character string (default NULL). DBnomics code of one or multiple datasets of a provider. If NULL, the datasets codes are downloaded with the function rdb_datasets and then the dimensions are requested.

use_readLines  Logical (default FALSE). If TRUE, then the data are requested and read with the base function readLines i.e. through the default R internet connection. This can be used to get round the error Could not resolve host: api.db.nomics.world.

curl_config  Named list (default NULL). If not NULL, it is used to configure a proxy connection. This configuration is passed to the function curl_fetch_memory of the package curl. A temporary curl_handle object is created internally with arguments equal to the provided list in curl_config. For curl_fetch_memory arguments see curl_fetch. For available curl options see curl_options, names(curl_options()) and libcurl.

simplify  Logical (default FALSE). If TRUE, when the dimensions are requested for only one provider and one dataset then a named list of data.tables is returned, not a nested named list of data.tables.

...  Additionals arguments.
Details

By default, the function returns a nested named list of `data.table`s containing the dimensions of datasets for providers from DBnomics.

Value

A nested named list of `data.table`s or a named list of `data.table`s.

Author(s)

Sebastien Galais

See Also

`rdb_providers`, `rdb_last_updates`, `rdb_datasets`, `rdb_series`

Examples

```r
## Not run:
rdb_dimensions(provider_code = "IMF", dataset_code = "WEO:2019-10")

rdb_dimensions(provider_code = "IMF", dataset_code = "WEO:2019-10", simplify = TRUE)

rdb_dimensions(provider_code = "IMF")

# !\ It is very long !
options(rdbnomics.progress_bar_dimensions = TRUE)
rdb_dimensions()
options(rdbnomics.progress_bar_dimensions = FALSE)

rdb_dimensions(
  provider_code = "IMF", dataset_code = "WEO:2019-10",
  use_readLines = TRUE
)

rdb_dimensions(
  provider_code = "IMF", dataset_code = "WEO:2019-10",
  curl_config = list(proxy = "<proxy>", proxyport = <port>)
)

## End(Not run)
```

rdb_last_updates

Download informations about the last DBnomics updates.

Description

`rdb_last_updates` downloads informations about the last updates from DBnomics.
Usage

```r
rdb_last_updates(
  all = FALSE,
  use_readLines = getOption("rdbnomics.use_readLines"),
  curl_config = getOption("rdbnomics.curl_config")
)
```

Arguments

- **all** Logical (default FALSE). If TRUE, then the full dataset of the last updates is retrieved.
- **use_readLines** Logical (default FALSE). If TRUE, then the data are requested and read with the base function `readLines` i.e. through the default R internet connection. This can be used to get round the error `Could not resolve host: api.db.nomics.world`.
- **curl_config** Named list (default NULL). If not NULL, it is used to configure a proxy connection. This configuration is passed to the function `curl_fetch_memory` of the package `curl`. A temporary `curl_handle` object is created internally with arguments equal to the provided list in `curl_config`. For `curl_fetch_memory` arguments see `curl_fetch`. For available curl options see `curl_options`, `names(curl_options())` and `libcurl`.

Details

By default, the function returns a `data.table` containing the last 100 updates from DBnomics with additional informations.

Value

A `data.table`.

Author(s)

Sebastien Galais

See Also

- `rdb_providers`, `rdb_datasets`, `rdb_dimensions`

Examples

```r
## Not run:
rdb_last_updates()
rdb_last_updates(all = TRUE)
rdb_last_updates(use_readLines = TRUE)
rdb_last_updates(curl_config = list(proxy = "<proxy>", proxyport = <port>))
## End(Not run)```
rdb_providers

Download list of DBnomics providers.

Description

rdb_providers downloads the list of providers from DBnomics.

Usage

rdb_providers(
  code = FALSE,
  use_readLines = getOption("rdbnomics.use_readLines"),
  curl_config = getOption("rdbnomics.curl_config")
)

Arguments

code Logical (default FALSE). If TRUE, then only the providers are returned in a vector.

use_readLines Logical (default FALSE). If TRUE, then the data are requested and read with the base function readLines i.e. through the default R internet connection. This can be used to get round the error Could not resolve host: api.db.nomics.world.

curl_config Named list (default NULL). If not NULL, it is used to configure a proxy connection. This configuration is passed to the function curl_fetch_memory of the package curl. A temporary curl_handle object is created internally with arguments equal to the provided list in curl_config.

Details

By default, the function returns a data.table containing the list of providers from DBnomics with additional informations such as the region, the website, etc.

Value

A data.table or a vector.

Author(s)

Sebastien Galais

See Also

rdb_last_updates, rdb_datasets, rdb_dimensions, rdb_series
### Examples

```r
# Not run:
rdb_providers()

rdb_providers(code = TRUE)

rdb_providers(use_readLines = TRUE)

rdb_providers(curl_config = list(proxy = "<proxy>", proxyport = <port>))

# End(Not run)
```

### Description

In the `xts` object returned by the function `rdb_to_xts`, the series codes are used as column names. If you prefer the series names (or apply a function to them), the function `rdb_rename_xts` is here for that.

### Usage

```r
rdb_rename_xts(x, fun = NULL, ...)
```

### Arguments

- `x`: `xts` object. The `xts` object returned by the function `rdb_to_xts`.
- `fun`: function (default `NULL`). The function to apply to the column names.
- `...`: Arguments for the function `fun`.

### Value

A `xts` object.

### Author(s)

Sebastien Galais

### See Also

`rdb`, `rdb_to_xts`
rdb_series

Examples

```r
## Not run:
library(xts)
library(data.table)
library(rdbnomics)

df <- rdb("IMF", "BOP", mask = "A.FR+ES.BCA_BP6_EUR")
df <- rdb_to_xts(df)
rdb_rename_xts(df)

## End(Not run)
```

---

**rdb_series**  
*Download list of series for datasets of DBnomics providers.*

---

**Description**

rdb_series downloads the list of series for available datasets of a selection of providers from DBnomics.  

\*\*\* We warn the user that this function can be (very) long to execute. We remind that DBnomics requests data from 63 providers to retrieve 21675 datasets for a total of approximately 720 millions series.

**Usage**

```r
rdb_series(
  provider_code = NULL,
  dataset_code = NULL,
  dimensions = NULL,
  query = NULL,
  use_readLines = getOption("rdbnomics.use_readLines"),
  curl_config = getOption("rdbnomics.curl_config"),
  simplify = FALSE,
  verbose = FALSE,
  ...
)
```

**Arguments**

- **provider_code**: Character string (default NULL). DBnomics code of one or multiple providers. If NULL, the providers are firstly downloaded with the function `rdb_providers` and then the datasets are requested.

- **dataset_code**: Character string (default NULL). DBnomics code of one or multiple datasets of a provider. If NULL, the datasets codes are downloaded with the function `rdb_datasets` and then the series are requested.
### rdb_series

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimensions</td>
<td>List or character string (single quoted) (default NULL). DBnomics code of one or several dimensions in the specified provider and dataset. If it is a named list, then the function toJSON (from the package jsonlite) is applied to generate the json object.</td>
</tr>
<tr>
<td>query</td>
<td>Character string (default NULL). A query to filter/select series from a provider’s dataset.</td>
</tr>
<tr>
<td>use_readLines</td>
<td>Logical (default FALSE). If TRUE, then the data are requested and read with the base function readLines i.e. through the default R internet connection. This can be used to get round the error Could not resolve host: api.db.nomics.world.</td>
</tr>
<tr>
<td>curl_config</td>
<td>Named list (default NULL). If not NULL, it is used to configure a proxy connection. This configuration is passed to the function curl_fetch_memory of the package curl. A temporary curl_handle object is created internally with arguments equal to the provided list in curl_config. For curl_fetch_memory arguments see curl_fetch. For available curl options see curl_options, names(curl_options()) and libcurl.</td>
</tr>
<tr>
<td>simplify</td>
<td>Logical (default FALSE). If TRUE, when the series are requested for only one provider and one dataset then a data.table is returned, not a nested named list of data.tables.</td>
</tr>
<tr>
<td>verbose</td>
<td>Logical (default FALSE). Show number of series per datasets and providers.</td>
</tr>
</tbody>
</table>

#### Details

By default, the function returns a nested named list of data.tables containing the series of datasets for providers from DBnomics.

#### Value

A nested named list of data.tables or a data.table.

#### Author(s)

Sebastien Galais

#### See Also

rdb_providers, rdb_last_updates, rdb_datasets, rdb_dimensions

#### Examples

```r
## Not run:
rdb_series(provider_code = "IMF", dataset_code = "WEO:2019-10")

## With dimensions
rdb_series("IMF", "WEO:2019-10", dimensions = list("weo-country" = "AGO"))
rdb_series("IMF", "WEO:2019-10", dimensions = list("weo-subject" = "NGDP_RPCH"), simplify = TRUE)

## With query
```
rdb_to_xts

Transform the data.table object into a xts object

Description

For some analysis, it is more convenient to have a xts object instead of a data.table object.

Usage

rdb_to_xts(
  x,
  needed_columns = c("period", "series_code", "series_name", "value"),
  series_columns = c("series_code", "series_name")
)

Arguments

x data.table. The data.table returned by the rdb function.
needed_columns Vector of character strings (default c("period", "series_code", "series_name", "value")). Vector of column names which are needed to transform the data.table into a xts object.
series_columns Vector of character strings (default c("series_code", "series_name")). Vector of series column names.

Value

A xts object.
Author(s)
Sebastien Galais

See Also
rdb, rdb_rename_xts

Examples
```r
## Not run:
library(xts)
library(data.table)
library(rdbnomics)

df <- rdb("IMF", "BOP", mask = "A.FR+ES.BCA_BP6_EUR")
rdb_to_xts(df)

## End(Not run)
```
Index

curl_fetch, 4, 8, 11, 12, 14, 15, 18
curl_options, 4, 8, 11, 12, 14, 15, 18

dbnomincs, 2

rdb, 3, 8, 16, 20
rdb_by_api_link, 7
rdb_datasets, 10, 12–15, 17, 18
rdb_dimensions, 11, 12, 14, 15, 18
rdb_last_updates, 11, 13, 13, 15, 18
rdb_providers, 11–14, 15, 17, 18
rdb_rename_xts, 16, 20
rdb_series, 11, 13, 15, 17
rdb_to_xts, 16, 19
rdbnomics, 7