Package ‘rdbnomics’

February 5, 2020

Type Package
Title Download DBnomics Data
Version 0.5.2
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://github.com/dbnomics/rdbnomics
BugReports https://github.com/dbnomics/rdbnomics/issues
Encoding UTF-8
LazyData true
Imports curl, jsonlite, data.table
RoxygenNote 7.0.2
Suggests knitr, rmarkdown, dplyr, magrittr, ggplot2, DT, testthat
VignetteBuilder knitr
NeedsCompilation no
Author Sebastien Galais [cre, ctb],
Thomas Brand [aut]
Maintainer Sebastien Galais <s915.stem@gmail.com>
Repository CRAN
Date/Publication 2020-02-05 19:00:05 UTC

R topics documented:

dbnomics ................................................................. 2
rdb ................................................................. 3
rdbnomics ................................................................. 7
rdb_by_api_link ................................................................. 7
rdb_last_updates ............................................................... 10
rdb_providers ............................................................... 11
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 theme.

Author(s)

Sebastien Galais

Examples

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

rdb("IMF", "WEO", 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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>provider_code</td>
<td>Character string (default NULL). DBnomics code of the provider.</td>
</tr>
<tr>
<td>dataset_code</td>
<td>Character string (default NULL). DBnomics code of the dataset.</td>
</tr>
<tr>
<td>ids</td>
<td>Character string (default NULL). DBnomics code of one or several series.</td>
</tr>
<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>mask</td>
<td>Character string (default NULL). DBnomics code of one or several masks in the specified provider and dataset.</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>api_link</td>
<td>Character string. DBnomics API link of the search. It should starts with http:// or https://.</td>
</tr>
<tr>
<td>filters</td>
<td>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,</td>
</tr>
</tbody>
</table>

Download DBnomics data.
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**

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

*or*
```
df1 <- rdb("AMECO", "ZUTN", dimensions = '{"geo": ["ea12"]}')
```

*Fetch two values of one dimension from dataset 'Unemployment rate' (ZUTN) of AMECO provider:*
```
df2 <- rdb("AMECO", "ZUTN", dimensions = list(geo = c("ea12", "dnk")))
```

*or*
```
df2 <- rdb("AMECO", "ZUTN", dimensions = '{"geo": ["ea12", "dnk"]}')
```

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

*or*
```
dim <- paste0(
  '{"country": ["DZ", "PE"],',
  '"indicator": ["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:*
```
df1 <- rdb("IMF", "BOP", mask = "A.FR.BCA_BP6_EUR")
```

*or when no argument names are given except provider_code and dataset_code (ids -> mask)*
```
df1 <- rdb("IMF", "BOP", "A.FR.BCA_BP6_EUR")
```

*Fetch two series from dataset 'Balance of Payments' (BOP) of IMF:*
```
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:*
```
df3 <- rdb("IMF", "BOP", mask = "A..BCA_BP6_EUR")
```

*Fetch series along multiple dimensions from dataset 'Balance of Payments' (BOP) of IMF:*
```
df4 <- rdb("IMF", "BOP", mask = "A.FR.BCA_BP6_EUR+IA_BP6_EUR")
```

## By query

*Fetch one series from dataset 'WEO by countries' (WEO) from IMF:*
```
df1 <- rdb("IMF", "WEO", query = "France current account balance percent")
```

*Fetch series from dataset 'WEO by countries' (WEO) from IMF:*
```
df2 <- rdb("IMF", "WEO", query = "current account balance percent")
```

## By api_link

*Fetch two series from different datasets of different providers:*
```
df1 <- rdb(
  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(
  api_link = paste0(
    "https://api.db.nomics.world/v22/series/WB/DB?dimensions=%7B%22indicator%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(
  ids = c("IMF/WEO/ABW.BCA", "IMF/WEO/ABW.BCA_NGDPO"),
  filters = list(
    code = "interpolate",
    parameters = list(frequency = "daily", method = "spline")
  )
)
```r
# Two filters
df1 <- rdb(
  ids = c("IMF/WEO/ABW.BCA", "IMF/WEO/ABW.BCA_NGDPD"),
  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). The API link is given on the DBnomics website.

Value

A data.table.

Author(s)

Sebastien Galais

See Also

rdb

Examples

## 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(

```r
## 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%22indicator%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 to use once

df2 <- rdb_by_api_link(
  paste0("https://api.db.nomics.world/v22/series/WB/DB?dimensions=%7B%22indicator%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"
  ),
  curl_config = h
)

## 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%22indicator%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 to use once

df2 <- rdb_by_api_link(
  paste0("https://api.db.nomics.world/v22/series/WB/DB?dimensions=%7B%22indicator%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"
  ),
  use_readLines = TRUE
)
```
## Apply filter(s) to the series

# One filter
df3 <- rdb_by_api_link(
  "https://api.db.nomics.world/v22/series/IMF/WEO/ABW.BCA?observations=1",
  filters = list(
    code = "interpolate",
    parameters = list(frequency = "daily", method = "spline")
  )
)

# Two filters
df3 <- rdb_by_api_link(
  "https://api.db.nomics.world/v22/series/IMF/WEO/ABW.BCA?observations=1",
  filters = list(
    list(
      code = "interpolate",
      parameters = list(frequency = "quarterly", method = "spline")
    ),
    list(
      code = "aggregate",
      parameters = list(frequency = "annual", method = "average")
    )
  )
)

## 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

Examples

## 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.
rdb Providers

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. 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 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

Examples

## 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)
Index

curl_fetch, 4, 8, 11, 12
curl_options, 4, 8, 11, 12

dbnomics, 2

rdb, 3, 8
rdb_by_api_link, 7
rdb_last_updates, 10, 12
rdb_providers, 11, 11
rdbnomics, 7

theme, 2