DBnomics : the world's economic database

Explore all the economic data from different providers (national and international statistical institutes, central banks, etc.), for free, following the link db.nomics.world.

You can also retrieve all the economic data through the `rdbnomics` package here. This blog post describes the different ways to do so.

Fetch time series by `ids`

First, let's assume that we know which series we want to download. A series identifier (`ids`) is defined by three values, formatted like this: `provider_code/dataset_code/series_code`.

Fetch one series from dataset 'Unemployment rate' (ZUTN) of AMECO provider

```
library(magrittr)
library(dplyr)
library(ggplot2)
library(rdbnomics)

df <- rdb(ids = 'AMECO/ZUTN/EA19.1.0.0.0.ZUTN') %>%
  filter(!is.na(value))
```

In such data.frame (data.table or tibble), you will always find at least nine columns:

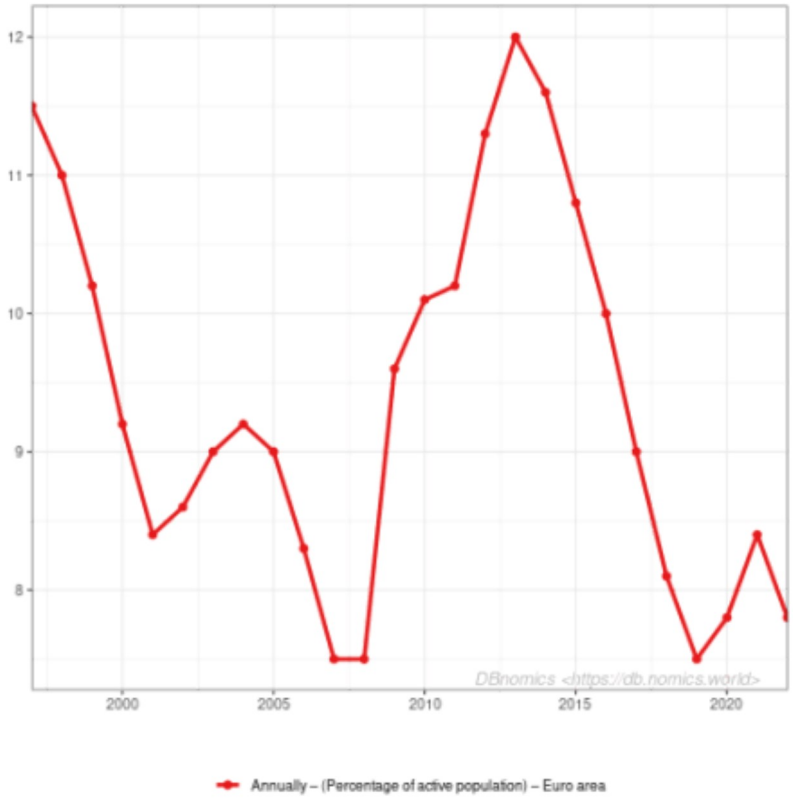
- `provider_code`
- `dataset_code`
- `dataset_name`
- `series_code`
- `series_name`
- `original_period` (character string)
- `period` (date of the first day of `original_period`)
- `original_value` (character string)
- `value`
- `@frequency` (harmonized frequency generated by DBnomics)

The other columns depend on the provider and on the dataset. They always come in pairs (for the code and the name). In the data.frame `df`, you have:

- `unit` (code) and `Unit` (name)
- `geo` (code) and `Country` (name)
- `freq` (code) and `Frequency` (name)

plot of chunk unnamed-chunk-5

```
ggplot(df, aes(x = period, y = value, color = series_name)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  dbnomics()
```



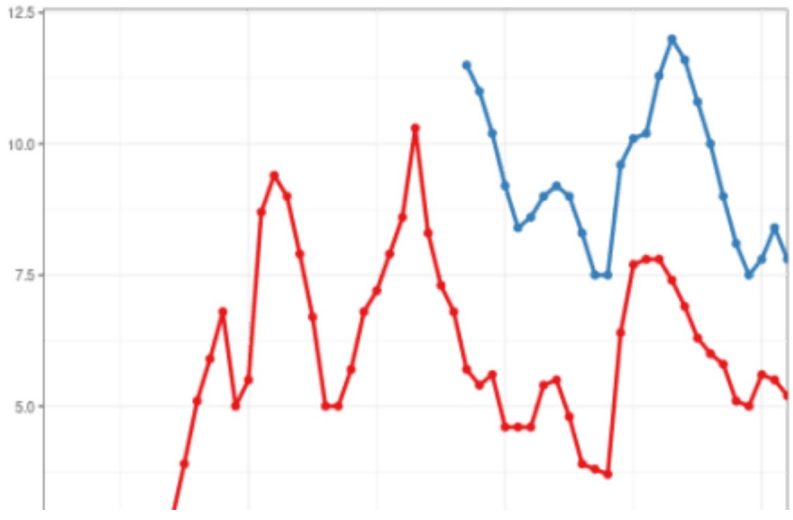
In the event that you only use the argument `ids`, you can drop it and run:

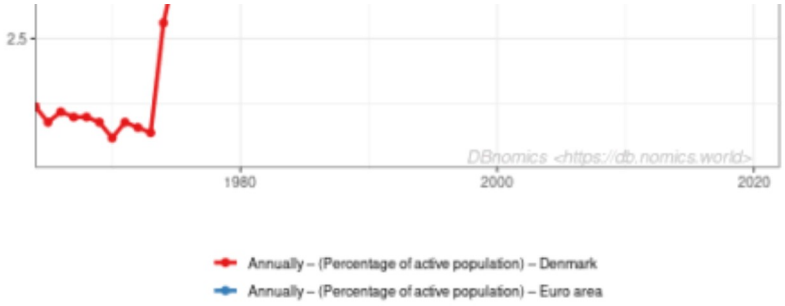
Fetch two series from dataset ‘Unemployment rate’ (ZUTN) of AMECO provider

```
df <- rdb(ids = c('AMECO/ZUTN/EA19.1.0.0.0.ZUTN', 'AMECO/ZUTN/DNK.1.0.0.0.ZUTN')
) %>%
  filter(!is.na(value))
```

plot of chunk unnamed-chunk-9

```
ggplot(df, aes(x = period, y = value, color = series_name)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  dbnomics()
```



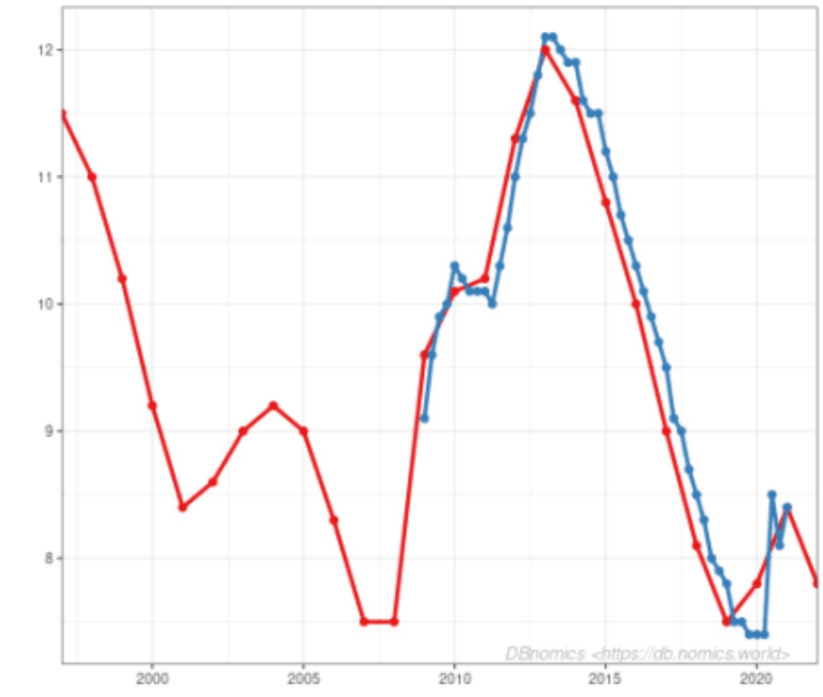


Fetch two series from different datasets of different providers

```
df <- rdb(ids = c('AMECO/ZUTN/EA19.1.0.0.0.ZUTN', 'Eurostat/une_rt_q/Q.SA.TOTAL.PC_ACT.T.EA19')) %>%
  filter(!is.na(value))
```

plot of chunk unnamed-chunk-12

```
ggplot(df, aes(x = period, y = value, color = series_name)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  dbnomics() +
  theme(legend.text = element_text(size=7))
```



nually – (Percentage of active population) – Euro area

uarterly – Seasonally adjusted data, not calendar adjusted data – From 15 to 74 years – Percentage of population in the labour force – Total – Euro area - 19 co

Fetch time series by `mask`

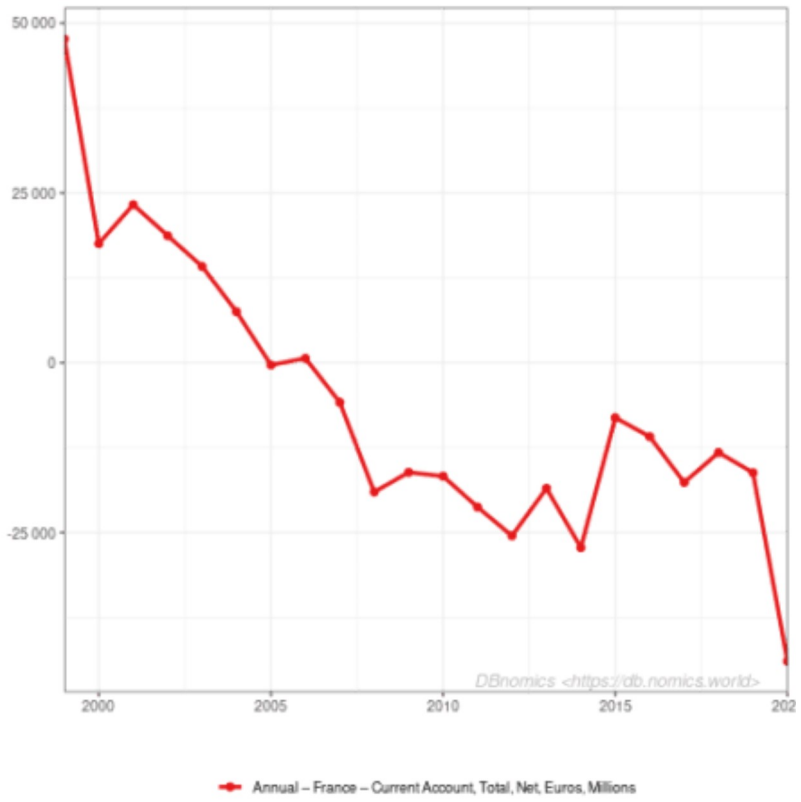
The code mask notation is a very concise way to select one or many time series at once. It is compatible only with some providers : BIS, ECB, Eurostat, FED, ILO, IMF, INSEE, OECD, WTO.

Fetch one series from dataset ‘Balance of Payments’ (BOP) of IMF

```
df <- rdb('IMF', 'BOP', mask = 'A.FR.BCA_BP6_EUR') %>%
  filter(!is.na(value))
```

plot of chunk unnamed-chunk-15

```
ggplot(df, aes(x = period, y = value, color = series_name)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  dbnomics()
```



In the event that you only use the arguments `provider_code`, `dataset_code` and `mask`, you can drop the name `mask` and run:

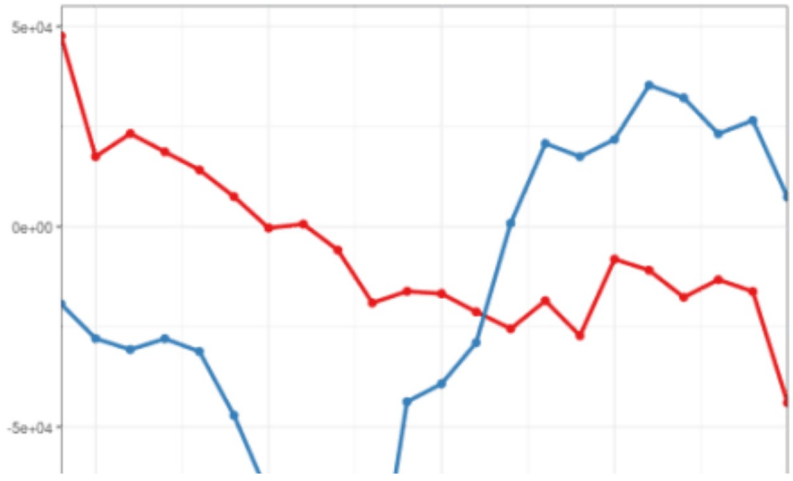
Fetch two series from dataset ‘Balance of Payments’ (BOP) of IMF

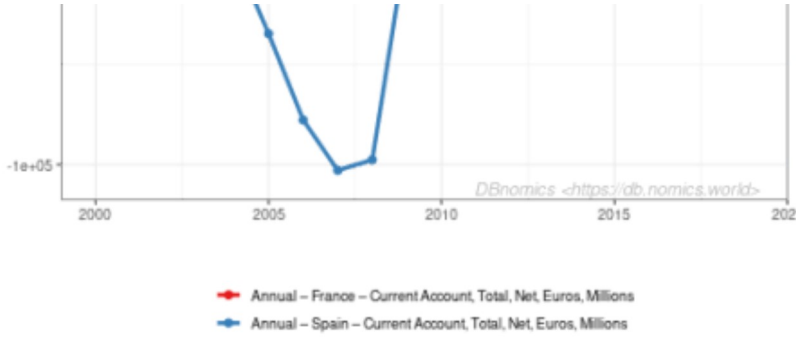
You just have to add a `+` between two different values of a dimension.

```
df <- rdb('IMF', 'BOP', mask = 'A.FR+ES.BCA_BP6_EUR') %>%
  filter(!is.na(value))
```

plot of chunk unnamed-chunk-19

```
ggplot(df, aes(x = period, y = value, color = series_name)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  dbnomics()
```





Fetch all series along one dimension from dataset ‘Balance of Payments’ (BOP) of IMF

```
df <- rdb('IMF', 'BOP', mask = 'A..BCA_BP6_EUR') %>%
  filter(!is.na(value)) %>%
  arrange(desc(period), REF_AREA) %>%
  head(100)
```

plot of chunk unnamed-chunk-22

Fetch series along multiple dimensions from dataset ‘Balance of Payments’ (BOP) of IMF

```
df <- rdb('IMF', 'BOP', mask = 'A.FR.BCA_BP6_EUR+IA_BP6_EUR') %>%
  filter(!is.na(value)) %>%
  group_by(INDICATOR) %>%
  top_n(n = 50, wt = period)
```

plot of chunk unnamed-chunk-24

Fetch time series by `dimensions`

Searching by `dimensions` is a less concise way to select time series than using the code `mask`, but it works with all the different providers. You have a “*Description of series code*” at the bottom of each dataset page on the DBnomics website.

Fetch one value of one dimension from dataset ‘Unemployment rate’ (ZUTN) of AMECO provider

```
df <- rdb('AMECO', 'ZUTN', dimensions = list(geo = "ea19")) %>%
  filter(!is.na(value))
# or
# df <- rdb('AMECO', 'ZUTN', dimensions = '{"geo": ["ea19"]}') %>%
#   filter(!is.na(value))
```

plot of chunk unnamed-chunk-26

```
ggplot(df, aes(x = period, y = value, color = series_name)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  dbnomics()
```

plot of chunk unnamed-chunk-27

Fetch two values of one dimension from dataset ‘Unemployment rate’ (ZUTN) of AMECO provider

```
df <- rdb('AMECO', 'ZUTN', dimensions = list(geo = c("ea19", "dnk"))) %>%
  filter(!is.na(value))
```

```
# or
# df <- rdb('AMECO', 'ZUTN', dimensions = '{"geo": ["ea19", "dnk"]}') %>%
#   filter(!is.na(value))
```

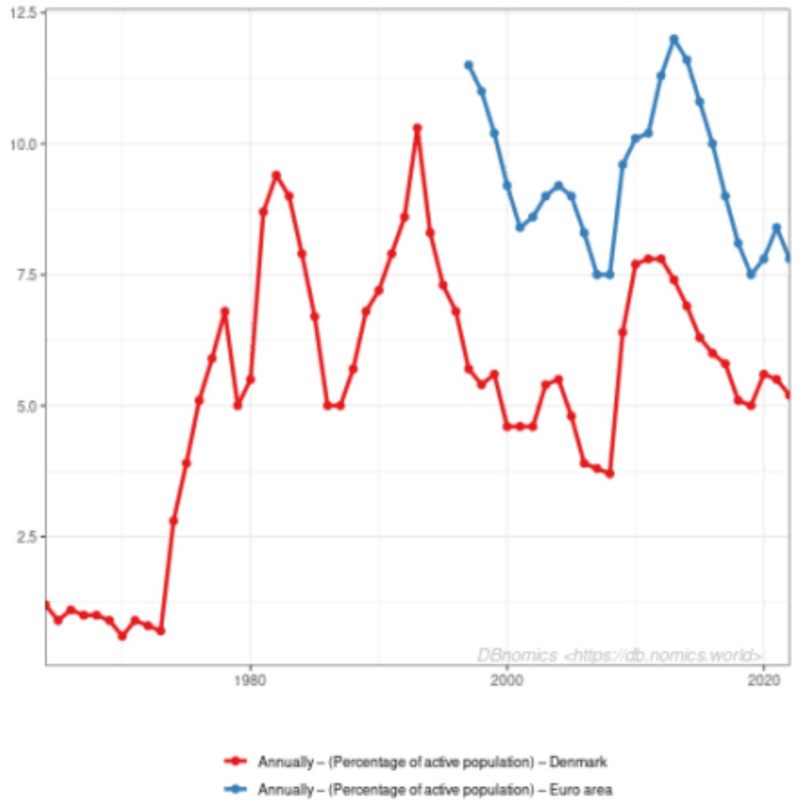
plot of chunk unnamed-chunk-29

```
ggplot(df, aes(x = period, y = value, color = series_name)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  dbnomics()
```

plot of chunk unnamed-chunk-30

Fetch several values of several dimensions from dataset ‘Doing business’ (DB) of World Bank

```
df <- rdb('WB', 'DB', dimensions = list(country = c("DZ", "PE"), indicator =
c("ENF.CONT.COEN.COST.ZS", "IC.REG.COST.PC.FE.ZS"))) %>%
  filter(!is.na(value))
# or
# df <- rdb('WB', 'DB', dimensions = '{"country": ["DZ", "PE"], "indicator":
["ENF.CONT.COEN.COST.ZS", "IC.REG.COST.PC.FE.ZS"]}') %>%
#   filter(!is.na(value))
```




```
ggplot(df, aes(x = period, y = value, color = series_name)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  dbnomics()
```

plot of chunk unnamed-chunk-33

Fetch time series found on the web site

When you don’t know the codes of the dimensions, provider, dataset or series, you can:

- go to the page of a dataset on DBnomics website, for example Doing Business,
- select some dimensions by using the input widgets of the left column,

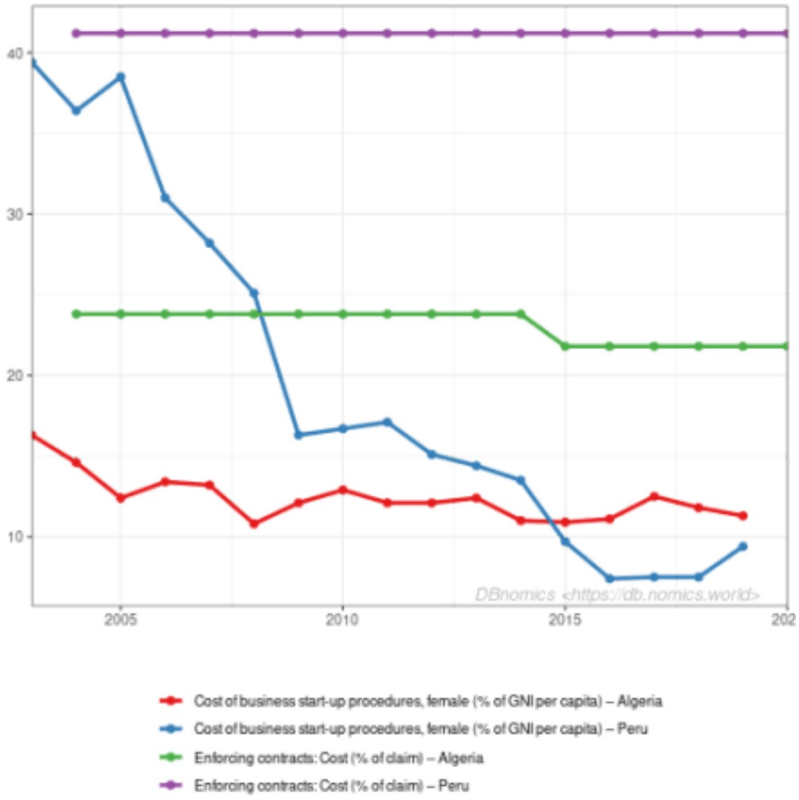
- click on "*Copy API link*" in the menu of the "*Download*" button,

- use the `rdb_by_api_link` function such as below.

```
df <- rdb_by_api_link("https://api.db.nomics.world/v22/series/WB/DB?dimensions=
%7B%22country%22%3A%5B%22FR%22%2C%22IT%22%2C%22ES%22%5D%7D&q=IC.REG.
PROC.FE.NO&observations=1&format=json&align_periods=1&offset=0&facets=0") %>%
  filter(!is.na(value))
```

plot of chunk unnamed-chunk-35

```
ggplot(df, aes(x = period, y = value, color = series_name)) +
  geom_step(size = 1.2) +
  geom_point(size = 2) +
  dbnomics()
```



Fetch time series from the cart

On the cart page of the DBnomics website, click on "*Copy API link*" and copy-paste it as an argument of the `rdb_by_api_link` function. Please note that when you update your cart, you have to copy this link again, because the link itself contains the ids of the series in the cart.

```
df <- rdb_by_api_link("https://api.db.nomics.world/v22/series?observations=1&series_ids=BOE/
6008/RPMTDDC,BOE/6231/RPMTBVE") %>%
```

```
  filter(!is.na(value))
```


```
ggplot(df, aes(x = period, y = value, color = series_name)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  dbnomics()
```

plot of chunk unnamed-chunk-40

Proxy configuration or connection error `Could not resolve host`

When using the functions `rdb` or `rdb_...`, you may come across the following error:

```
Error in open.connection(con, "rb") :
  Could not resolve host: api.db.nomics.world
```

To get round this situation, you have two options:

1. configure **curl** to use a specific and authorized proxy.
2. use the default R internet connection i.e. the Internet Explorer proxy defined in *internet2.dll*.

Configure **curl** to use a specific and authorized proxy

In **rdbnomics**, by default the function `curl_fetch_memory` (of the package **curl**) is used to fetch the data. If a specific proxy must be used, it is possible to define it permanently with the package option `rdbnomics.curl_config` or on the fly through the argument `curl_config`. Because the object is a named list, its elements are passed to the connection (the `curl_handle` object created internally with `new_handle()`) with `handle_setopt()` before using `curl_fetch_memory`.

To see the available parameters, run `names(curl_options())` in *R* or visit the website https://curl.haxx.se/libcurl/c/curl_easy_setopt.html. Once they are chosen, you define the curl object as follows:

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

Set the connection up for a session

The curl connection can be set up for a session by modifying the following package option:

```
options(rdbnomics.curl_config = h)
```

When fetching the data, the following command is executed:

```
hndl <- curl::new_handle()
curl::handle_setopt(hndl, .list = getOption("rdbnomics.curl_config"))
curl::curl_fetch_memory(url = <...>, handle = hndl)
```

After configuration, just use the standard functions of **rdbnomics** e.g.:

```
df1 <- rdb(ids = 'AMECO/ZUTN/EA19.1.0.0.0.ZUTN')
```

This option of the package can be disabled with:

```
options(rdbnomics.curl = NULL)
```

Use the connection only for a function call

If a complete configuration is not needed but just an “on the fly” execution, then use the argument `curl_config` of the functions `rdb` and `rdb_...`:

```
df1 <- rdb(ids = 'AMECO/ZUTN/EA19.1.0.0.0.ZUTN', curl_config = h)
```

Use the default R internet connection

To retrieve the data with the default R internet connection, **rdbnomics** will use the base function `readLines`.

Set the connection up for a session

To activate this feature for a session, you need to enable an option of the package :

```
options(rdbnomics.use_readLines = TRUE)
```

And then use the standard function as follows :

```
df1 <- rdb(ids = 'AMECO/ZUTN/EA19.1.0.0.0.ZUTN')
```

This configuration can be disabled with :

```
options(rdbnomics.use_readLines = FALSE)
```

Use the connection only for a function call

If you just want to do it once, you may use the argument `use_readLines` of the functions `rdb` and `rdb_...` :

```
df1 <- rdb(ids = 'AMECO/ZUTN/EA19.1.0.0.0.ZUTN', use_readLines = TRUE)
```