Package ‘cbsodataR’

January 8, 2024

Type     Package
Title    Statistics Netherlands (CBS) Open Data API Client
Version  1.0.1
Description The data and meta data from Statistics Netherlands (<https://www.cbs.nl>) can be browsed and downloaded. The client uses the open data API of Statistics Netherlands.
License  GPL-2
URL      https://github.com/edwindj/cbsodataR
BugReports https://github.com/edwindj/cbsodataR/issues
Encoding UTF-8
Imports  whisker, jsonlite, utils
Suggests knitr, rmarkdown, dplyr, shiny, testthat (>= 2.1.0), sf
VignetteBuilder knitr
RoxygenNote 7.2.3
NeedsCompilation no
Author   Edwin de Jonge [aut, cre],
         Sara Houweling [ctb]
Maintainer Edwin de Jonge <edwindjonge@gmail.com>
Repository CRAN
Date/Publication 2024-01-08 15:40:02 UTC

R topics documented:

  cbsodataR-package 2
  cache_clear 4
  cbs_add_date_column 4
  cbs_add_label_columns 5
  cbs_add_statcode_column 6
  cbs_default_selection 8
  cbs_download_data 8
R package

Download all data from Statistics Netherlands / CBS

cbsodataR allows to download all official statistics of Statistics Netherlands (CBS) into R. For an introduction please visit the vignette: vignette("cbsodataR", package="cbsodataR"). For an introduction on using CBS cartographic maps: vignette("maps", package="cbsodataR"). The functions cbs_get_datasets() and cbs_get_data() should get you going. Interested in cartographic maps, see cbs_get_maps().

Catalog function

- **cbs_get_datasets()** returns a data.frame with table of contents (toc): the publication meta data for available tables, can also include the extra tables not directly available in StatLine (dataderden)
- **cbs_get_catalogs()**, returns data.frame with the available (extra) catalogs.
• `cbs_get_toc()`, returns a data.frame with table of contents (toc): the publication meta data for available tables within the standard CBS
• `cbs_search()`, returns a data.frame with tables that contain the given search word.

Data retrieval

• `cbs_get_data()`, returns the data of a specific opendata/StatLine table
• `cbs_download_table()`, saves the data (and metadata) as csv files into a directory

Meta data

• `cbs_get_meta()`, returns the meta data objects of a specific opendata / StatLine table.
• `cbs_add_date_column()`, converts date/period codes into DateTime objects in the data set that was downloaded.
• `cbs_add_label_columns()`, adds labels to the code columns in the data that was downloaded.

Cartographic maps

• `cbs_get_maps()`, returns a data.frame with available CBS maps
• `cbs_join_sf_with_data()`, returns an sf object joined with cbs table
• `cbs_get_sf()`, returns an sf object without data, e.g. "gemeente_2020".

Copyright use

The content of CBS opendata is subject to Creative Commons Attribution (CC BY 4.0). This means that the re-use of the content is permitted, provided Statistics Netherlands is cited as the source. For more information see: [https://www.cbs.nl/en-gb/about-us/website/copyright](https://www.cbs.nl/en-gb/about-us/website/copyright)

Author(s)

Maintainer: Edwin de Jonge <edwindjonge@gmail.com>

Other contributors:

• Sara Houweling [contributor]

See Also

Useful links:

• [https://github.com/edwindj/cbsodataR](https://github.com/edwindj/cbsodataR)
• Report bugs at [https://github.com/edwindj/cbsodataR/issues](https://github.com/edwindj/cbsodataR/issues)
cache_clear  

** cache_clear clears the cache

---

**cbs_add_date_column**  

Convert the time variable into either a date or numeric.

**Description**

Time periods in data of CBS are coded: yyyyXXww (e.g. 2018J00, 2018MM10, 2018KW02), which contains year (yyyy), type (XX) and index (ww). cbs_add_date_column converts these codes into a Date() or numeric. In addition it adds a frequency column denoting the type of the column.

**Usage**

```r
cbs_add_date_column(x, date_type = c("Date", "numeric"), ...)
```

**Arguments**

- `x`  
  data.frame retrieved using `cbs_get_data()`
- `date_type`  
  Type of date column: "Date", "numeric". Numeric creates a fractional number which signs the "middle" of the period. e.g. 2018J00 -> 2018.5 and 2018KW01 -> 2018.167. This is for the following reasons: otherwise 2018.0 could mean 2018, 2018 Q1 or 2018 Jan, and furthermore 2018.75 is a bit strange for 2018 Q4. If all codes in the dataset have frequency "Y" the numeric output will be integer.
- `...`  
  future use.

**Value**

original dataset with two added columns: `<period>_date` and `<period>_freq`. This last column is a factor with levels: Y, Q and M

**See Also**

Other data retrieval: `cbs_add_label_columns()`, `cbs_download_data()`, `cbs_extract_table_id()`, `cbs_get_data_from_link()`, `cbs_get_data()`

Other meta data: `cbs_add_label_columns()`, `cbs_download_meta()`, `cbs_get_meta()`


### Examples

```r
## Not run:
x <- cbs_get_data( id = "7196ENG" # table id
, Periods = "2000MM03" # March 2000
, CPI = "000000" # Category code for total )

# add a Periods_Date column
x <- cbs_add_date_column(x)

# add a Periods_numeric column
x <- cbs_add_date_column(x, date_type = "numeric")

## End(Not run)
```

---

**cbs_add_label_columns**  
*For each column with codes add label column to data set*

#### Description

Adds cbs labels to the dataset that was retrieved using `cbs_get_data()`.

#### Usage

```r
cbs_add_label_columns(x, columns = colnames(x), ...)
```

#### Arguments

- `x`  
  data.frame retrieved using `cbs_get_data()`.

- `columns`  
  character with the names of the columns for which labels will be added

- `...`  
  not used.

#### Details

Code columns will be translated into label columns for each of the column that was supplied.

By default all code columns will be accompanied with a label column. The name of each label column will be `<code_column>_label`.

#### Value

the original data.frame `x` with extra label columns. (see description)

#### See Also

Other data retrieval: `cbs_add_date_column()`, `cbs_download_data()`, `cbs_extract_table_id()`, `cbs_get_data_from_link()`, `cbs_get_data()`

Other meta data: `cbs_add_date_column()`, `cbs_download_meta()`, `cbs_get_meta()`
cbs_add_statcode_column

Prepares dataset for making a map

Description

Adds a statcode column to the dataset, so it can be more easily joined with a map retrieved with \texttt{cbs_get_sf()}.  

Usage

\begin{verbatim}
cbs_add_statcode_column(x, ...)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{x} \hspace{1cm} data.frame retrieved using \texttt{cbs_get_data()}
\item \texttt{...} \hspace{1cm} future use.
\end{itemize}

Details

Regional data uses the \texttt{x$RegioS} dimension for data. The "codes" for each region are also used in the cartographic map boundaries of regions as used in \texttt{cbs_get_sf()}. Unfortunately the codes in \texttt{x$RegioS} can have trailing spaces, and the variable used in the mapping material is named \texttt{statcode}. This method simply adds a statcode column with trimmed codes from \texttt{RegioS}, making it more easy to connect a dataset to a cartographic map.

Value

original dataset with added statcode column.

See Also

Other cartographic map: \texttt{cbs_get_maps()}, \texttt{cbs_get_sf()}, \texttt{cbs_join_sf_with_data()}
Examples

```r
if (interactive()){

    # retrieve maps
cbs_maps <- cbs_get_maps()
cbs_maps |> head(4)

    gemeente_map <- cbs_get_sf("gemeente", 2023, verbose=TRUE)
    # sf object
gemeente_map

    # plot the statcodes (included in the map)
    plot(gemeente_map, max.plot = 1)

    # now connect with some data
    labor <- cbs_get_data("85268NED"
                        , Perioden = "2022JJ00" # only 2022
                        , RegioS = has_substring("PV") # only province
                        , verbose = TRUE
                        )

    # most conveniently
    provincie_2022_with_data <- cbs_join_sf_with_data("provincie", 2022, labor)

    # better plotting options are ggplot2 or tmap,
    # but keeping dependencies low...
    provincie_2022_with_data |> subset(select = Werkloosheidspercentage_13) |> plot( border ="#FFFFFF99", main="unemployment rate")

    ## but of course this can also be done by hand:
labor <- labor |> cbs_add_statcode_column() # add column to connect with map

    provincie_2022 <- cbs_get_sf("provincie", 2022)

    # this is a left_join(provincie_2022, labor, by = "statcode")
    provincie_2022_data <- within(provincie_2022, {
        unemployment_rate <- labor$Werkloosheidspercentage_13[match(statcode, labor$statcode)]
    })

    # better plotting options are ggplot2 or tmap,
    # but keeping dependencies low...
    plot( provincie_2022_data[,c("unemployment_rate")]
          , border ="#FFFFFF99"
          , nbreaks = 12
    )
}
```
**cbs_default_selection**  
*extract the default selection from a cbsodata meta object*

**Description**
extract the default selection from a cbsodata meta object

**Usage**
```r
cbs_default_selection(x, ...)
```

**Arguments**
- `x`  
  meta object
- `...`  
  for future use

---

**cbs_download_data**  
*Gets all data from a cbs table.*

**Description**
Gets all data via bulk download. `cbs_download_data` dumps the data in (international) csv format.

**Usage**
```r
cbs_download_data(
  id,
  path = file.path(id, "data.csv"),
  catalog = catalog,
  ...,  
  select = NULL,
  typed = TRUE,
  verbose = FALSE,
  show_progress = interactive() && !verbose,
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)
```

**Arguments**
- `id`  
  of cbs open data table
- `path`  
  of data file, defaults to "id/data.csv"
- `catalog`  
  catalog id, can be retrieved with `cbs_get_datasets()`
- `...`  
  optional filter statements to select rows of the data,
- `select`  
  optional names of columns to be returned.
cbs_download_meta

Dumps the meta data into a directory

description

Dumps the meta data into a directory

Usage

cbs_download_meta(
  id,
  dir = id,
  catalog = "CBS",
  ..., verbose = FALSE,
  cache = FALSE,
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)

Arguments

id Id of CBS open data table (see cbs_get_toc())
dir Directory in which data should be stored. By default it creates a sub directory with the name of the id
catalog catalog id, can be retrieved with cbs_get_datasets()
... not used
verbose Print extra messages what is happening.
cache Should meta data be cached?
base_url optionally allow to specify a different server. Useful for third party data services implementing the same protocol.
**Value**

meta data object

**See Also**

Other meta data: `cbs_add_date_column()`, `cbs_add_label_columns()`, `cbs_get_meta()`

Other download: `cbs_download_data()`, `cbs_download_table()`

---

**cbs_download_table**  
*Download a table from statistics Netherlands*

---

**Description**

`cbs_download_table` downloads the data and metadata of a table from statistics Netherlands and stores it in csv format.

**Usage**

```r
  cbs_download_table(
    id, 
    catalog = "CBS", 
    ..., 
    dir = id, 
    cache = FALSE, 
    verbose = TRUE, 
    typed = FALSE, 
    base_url = getOption("cbsodataR.base_url", BASE_URL)
  )
```

**Arguments**

- **id**  
  Identifier of CBS table (can be retrieved from `cbs_get_toc()`)
- **catalog**  
  catalog id, can be retrieved with `cbs_get_datasets()`
- **...**  
  Parameters passed on to `cbs_download_data()`
- **dir**  
  Directory where table should be downloaded
- **cache**  
  If metadata is cached use that, otherwise download meta data
- **verbose**  
  Print extra messages what is happening.
- **typed**  
  Should the data automatically be converted into integer and numeric?
- **base_url**  
  optionally specify a different server. Useful for third party data services implementing the same protocol.
Details

cbs_download_table retrieves all raw meta data and data and stores these as csv files in the directory specified by dir. It is possible to add a filter. A filter is specified with \(<\text{column\_name}\> = \langle\text{values}\rangle\) in which \(<\text{values}\rangle\) is a character vector. Rows with values that are not part of the character vector are not returned.

Value

meta data object of id `cbs_get_meta()`.

See Also

Other download: `cbs_download_data()`, `cbs_download_meta()`

Examples

```r
## Not run:

# download meta data and data from inflation/Consumer Price Indices
download_table(id="7196ENG")

## End(Not run)
```

---

cbs_extract_table_id  extract the id of a cbs table from the statline url

Description

extract the id of a cbs table from the statline url

Usage

cbs_extract_table_id(url, ...)

Arguments

- `url` character with hyperlink to StatLine table
- `...` future use.

Value

character with id, will be NA if not found.

See Also

Other data retrieval: `cbs_add_date_column()`, `cbs_add_label_columns()`, `cbs_download_data()`, `cbs_get_data_from_link()`, `cbs_get_data()`
cbs_get_catalogs

Description
Retrieves the possible catalog values that can be used for retrieving data

Usage
cbs_get_catalogs(..., base_url = BASE_URL)

Arguments
... filter statement to select rows, e.g. Language="nl"
base_url optionally specify a different server. Useful for third party data services implementing the same protocol.

Examples
if (interactive()){
catalogs <- cbs_get_catalogs()

# Identifier of catalog can be used to query
print(catalogs$Identifier)

d_s_rivm <- cbs_get_datasets(catalog = "RIVM")
d_s_rivm[1:5, c("Identifier","ShortTitle")]
}

cbs_get_data

Description
Get data from Statistics Netherlands (CBS)

Usage
cbs_get_data(
id,
...,
catalog = "CBS",
select = NULL,
cbs_get_data

typed = TRUE,
add_column_labels = TRUE,
dir = tempdir(),
verbose = FALSE,
base_url = getOption("cbsodataR.base_url", BASE_URL),
include_ID = FALSE
)

Arguments

id  Identifier of table, can be found in cbs_get_datasets()
...  optional filter statements, see details.
catalog  catalog id, can be retrieved with cbs_get_datasets() (set catalog=NULL to see all catalogs)
select  character optional, columns to select
typed  Should the data automatically be converted into integer and numeric?
add_column_labels  Should column titles be added as a label (TRUE) which are visible in View
dir  Directory where the table should be downloaded. Defaults to temporary directory
verbose  Print extra messages what is happening.
base_url  optionally specify a different server. Useful for third party data services implementing the same protocol.
include_ID  Should the data include the ID column for the rows?

Details
To reduce the download time, Optionally the data can be filtered on category values: for large tables (> 100k records) this is a wise thing to do.

The filter is specified with (see examples below):

• <column_name> = <values> in which <values> is a character vector. Rows with values that are not part of the character vector are not returned. Note that the values have to be values from the $Key column of the corresponding meta data. These may contain trailing spaces...

• <column_name> = has_substring(x) in which x is a character vector. Rows with values that do not have a substring that is in x are not returned. Useful substrings are "JJ", "KW", "MM" for Periods (years, quarters, months) and "PV", "CR" and "GM" for Regions (provinces, corops, municipalities).

• <column_name> = eq(<values>) | has_substring(x), which combines the two statements above.

By default the columns will be converted to their type (typed=TRUE). CBS uses multiple types of missing (unknown, supressed, not measured, missing): users wanting all these nuances can use typed=FALSE which results in character columns.
Value
data.frame with the requested data. Note that a csv copy of the data is stored in dir.

Copyright use
The content of CBS opendata is subject to Creative Commons Attribution (CC BY 4.0). This means that the re-use of the content is permitted, provided Statistics Netherlands is cited as the source. For more information see: https://www.cbs.nl/en-gb/about-us/website/copyright

Note
All data are downloaded using cbs_download_table()

See Also
cbs_get_meta(), cbs_download_data()
Other data retrieval: cbs_add_date_column(), cbs_add_label_columns(), cbs_download_data(), cbs_extract_table_id(), cbs_get_data_from_link()
Other query: eq(), has_substring()

Examples
## Not run:
cbs_get_data( id = "7196ENG" # table id
, Periods = "2000MM03" # March 2000
, CPI = "000000" # Category code for total
 )

# useful substrings:
## Periods: "JJ": years, "KW": quarters, "MM", months
## Regions: "NL", "PV": provinces, "GM": municipalities
cbs_get_data( id = "7196ENG" # table id
, Periods = has_substring("JJ") # all years
, CPI = "000000" # Category code for total
 )

cbs_get_data( id = "7196ENG" # table id
, Periods = c("2000MM03","2001MM12") # March 2000 and Dec 2001
, CPI = "000000" # Category code for total
 )

# combine either this
cbs_get_data( id = "7196ENG" # table id
, Periods = has_substring("JJ") | "2000MM01" # all years and Jan 2001
, CPI = "000000" # Category code for total
 )

# or this: note the "eq" function
cbs_get_data( id = "7196ENG" # table id
, Periods = eq("2000MM01") | has_substring("JJ") # Jan 2000 and all years
cbs_get_datasets

## End(Not run)

---

**cbs_get_datasets**

*Retrieve a data.frame with requested cbs tables*

### Description

cbs_get_datasets by default a list of all tables and all columns will be retrieved. You can restrict the query by supplying multiple filter statements or by specifying the columns that should be returned.

### Usage

cbs_get_datasets(
  catalog = "CBS",
  convert_dates = TRUE,
  select = NULL,
  verbose = FALSE,
  cache = TRUE,
  base_url = getOption("cbsodataR.base_url", BASE_URL),
  ...
)

### Arguments

- **catalog**: which set of tables should be returned? cbs_get_catalogs() or supply NULL for all tables.
- **convert_dates**: convert the columns with date-time information into DateTime (default TRUE)
- **select**: character columns to be returned, by default all columns will be returned.
- **verbose**: logical prints the calls to the webservice
- **cache**: logical should the result be cached?
- **base_url**: optionally specify a different server. Useful for third party data services implementing the same protocol.
- **...**: filter statement to select rows, e.g. Language="nl"

### Details

Note that setting `catalog` to NULL results in a datasets list with all tables including the extra catalogs.
Examples

```r
if (interactive()){
  # retrieve the datasets in the "CBS" catalog
  ds <- cbs_get_datasets()
  ds[1:5, c("Identifier", "ShortTitle")]

  # retrieve the datasets in the "AZW" catalog
  ds_azw <- cbs_get_datasets(catalog = "AZW")

  # to retrieve all datasets of all catalogs, supply "NULL"
  ds_all <- cbs_get_datasets(catalog = NULL)
}
```

cbs_get_data_from_link

*Retrieve data from a link created from the StatLine app.*

Description

Retrieve data from a link created from the StatLine app.

Usage

```r
cbs_get_data_from_link(
  link,
  message = TRUE,
  ..., 
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)
```

Arguments

- **link**: url/hyperlink to opendata table made with the StatLine App
- **message**: logical Should the query be printed (default TRUE)
- **...**: passed on to `cbs_get_data`
- **base_url**: optionally specify a different server. Useful for third party data services implementing the same protocol.

Value

Same as `cbs_get_data`

See Also

Other data retrieval: `cbs_add_date_column()`, `cbs_add_label_columns()`, `cbs_download_data()`, `cbs_extract_table_id()`.
cbs_get_maps

Get list of CBS maps

Description

Returns a list of (simplified) maps, that can be used with CBS data.

Usage

cbs_get_maps( verbose = FALSE, cache = TRUE )

Arguments

verbose  if TRUE a message with the download url will be printed.
cache   if TRUE the result will be cached.

Value

data.frame with region, year and links to geojson

See Also

Other cartographic map: cbs_add_statcode_column(), cbs_get_sf(), cbs_join_sf_with_data()

Examples

if (interactive()){

    # retrieve maps
cbs_maps <- cbs_get_maps()
cbs_maps |> head(4)

    gemeente_map <- cbs_get_sf("gemeente", 2023, verbose=TRUE)
    # sf object
    gemeente_map

    # plot the statcodes (included in the map)
    plot(gemeente_map, max.plot = 1)

    # now connect with some data
    labor <- cbs_get_data("85268NED"
        , Perioden = "2022JJ00" # only 2022
        , RegioS = has_substring("PV") # only province
        , verbose = TRUE
    )

    # most conveniently
    provincie_2022_with_data <- cbs_join_sf_with_data("provincie", 2022, labor)
}
# better plotting options are ggplot2 or tmap,
# but keeping dependencies low...
provincie_2022_with_data |> 
  subset(select = Werkloosheidspersentage_13) |> 
  plot( border = "#FFFFFF99", main = "unemployment rate")

## but of course this can also be done by hand:
labor <- labor |> 
  cbs_add_statcode_column() # add column to connect with map

provincie_2022 <- cbs_get_sf("provincie", 2022)

# this is a left_join(provincie_2022, labor, by = "statcode")
provincie_2022_data <- 
  within(provincie_2022, { 
    unemployment_rate <- labor$Werkloosheidspersentage_13[match(statcode, labor$statcode)]
  })

# better plotting options are ggplot2 or tmap,
# but keeping dependencies low...
plot( provincie_2022_data[, c("unemployment_rate")]
  , border = "#FFFFFF99"
  , nbreaks = 12
)

---
cbs_get_meta

Get metadata of a CBS table

Description

Retrieve the meta data of a CBS open data table. Caching (cache=TRUE) improves the performance considerably.

Usage

cbs_get_meta(
  id,
  catalog = "CBS",
  verbose = FALSE,
  cache = TRUE,
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)

Arguments

id internal id of CBS table, can be retrieved with cbs_get_datasets()
catalog catalog id, can be retrieved with cbs_get_datasets()
`cbs_get_meta_from_dir`  

**Verbose**  
Print extra messages what is happening.

**Cache**  
Should the result be cached?

**BaseUrl**  
Optionally specify a different server. Useful for third party data services implementing the same protocol.

**Details**  
The meta data of a CBS table is determined by the web API of Statistics Netherlands. `cbsodataR` stays close to this API. Each `cbsodataR` object has the following metadata items, which are all `data.frame`

- `$TableInfos`: data.frame with the descriptive publication metadata of the table, such as `Title`, `Description`, `Summary` etc.
- `$DataProperties`: data.frame with the `Title`, `Description`, `Unit` etc. of each column in the dataset that is downloaded with `cbs_get_data()`.
- `$CategoryGroups`: hierarchical groupings of the code columns.
- `$<code_column>`: for each code column a data.frame with the `Title`, `Key`, `Description` etc. of each code / category in that column. E.g. `Perioden` for time codes `c("2019JJ00","2018JJ00")`.

**Value**  
cbs_table object containing several `data.frame`s with meta data (see details)

**See Also**
- Other meta data: `cbs_add_date_column()`, `cbs_add_label_columns()`, `cbs_download_meta()`

---

**cbs_get_meta_from_dir**  
*Load meta data from a downloaded table*

**Description**  
Load meta data from a downloaded table

**Usage**  

```r
R> cbs_get_meta_from_dir(dir)
```

**Arguments**

- `dir`  
Directory where data was downloaded

**Value**  
cbs_table object with meta data
cbs_get_sf

Retrieve an sf map for plotting

Description
Retrieve a polygon sf object that can be used for plotting. This function only provides the region boundaries.

Usage
cbs_get_sf(
  region,
  year,
  keep_columns = c("statcode", "statnaam"),
  verbose = FALSE
)

Arguments
region character name of region
year integer year of a region
keep_columns character, set to NULL to retrieve all columns of the map
verbose if TRUE the method is verbose

Details
To use the map for plotting:
• add data columns to the sf data.frame returned by cbs_get_sf, e.g. by using dplyr::left_join or otherwise
• use ggplot2, tmap, leaflet or any other plotting library useful for plotting spatial data.

Value
sf::st_sf() object with the polygons of the regions specified.

See Also
Other cartographic map: cbs_add_statcode_column(), cbs_get_maps(), cbs_join_sf_with_data()

Examples
if (interactive()){  
  # retrieve maps
  cbs_maps <- cbs_get_maps()
  cbs_maps |> head(4)
gemeente_map <- cbs_get_sf("gemeente", 2023, verbose=TRUE)

# sf object
gemeente_map

# plot the statcodes (included in the map)
plot(gemeente_map, max.plot = 1)

# now connect with some data
labor <- cbs_get_data("85268NED"
  , Perioden = "2022JJ00" # only 2022
  , RegioS = has_substring("PV") # only province
  , verbose = TRUE
)

# most conveniently
provincie_2022_with_data <- cbs_join_sf_with_data("provincie", 2022, labor)

# better plotting options are ggplot2 or tmap,
# but keeping dependencies low...
provincie_2022_with_data |>
  subset(select = Werkloosheidspercentage_13) |>
  plot( border ="#FFFFFF99", main="unemployment rate")

## but of course this can also be done by hand:
labor <- labor |
  cbs_add_statcode_column() # add column to connect with map

provincie_2022 <- cbs_get_sf("provincie", 2022)

# this is a left_join(provincie_2022, labor, by = "statcode")
provincie_2022_data <-
  within(provincie_2022, {
    unemployment_rate <- labor$Werkloosheidspercentage_13[match(statcode, labor$statcode)]
  })

# better plotting options are ggplot2 or tmap,
# but keeping dependencies low...
plot( provincie_2022_data[,c("unemployment_rate")]
  , border ="#FFFFFF99"
  , nbreaks = 12
)

---

cbs_get_tables_themes  
*Get a the list of tables connected to themes*

**Description**
Get a the list of tables connected to themes
Usage

cbs_get_tables_themes(
    ..., 
    select = NULL, 
    verbose = FALSE, 
    cache = TRUE, 
    base_url = getOption("cbsodataR.base_url", BASE_URL)
)

Arguments

...  Use this to add a filter to the query e.g. get_tables_themes(ID=10).
select  character vector with names of wanted properties. default is all
verbose  Print extra messages what is happening.
cache  Should the result be cached?
base_url  optionally specify a different server. Useful for third party data services implementing the same protocol.

Value

A data.frame with various properties of SN/CBS themes.

---

cbs_get_themes  Get list of all cbs thematic entries.

Description

Returns a list of all cbs themes.

Usage

cbs_get_themes(
    ..., 
    select = NULL, 
    verbose = TRUE, 
    cache = FALSE, 
    base_url = getOption("cbsodataR.base_url", BASE_URL)
)

Arguments

...  Use this to add a filter to the query e.g. get_themes(ID=10).
select  character vector with names of wanted properties. default is all
verbose  Print extra messages what is happening.
cache  Should the result be cached?
base_url  optionally specify a different server. Useful for third party data services implementing the same protocol.
Value

A data.frame with various properties of SN/CBS themes.

The filter is specified with `<column_name> = <values>` in which `<values>` is a character vector. Rows with values that are not part of the character vector are not returned.

Examples

```r
## Not run:
# get list of all themes
cbs+get_themes()

# get list of all dutch themes from the Catalog "CBS"
cbs_get_themes(Language="nl", Catalog="CBS")

## End(Not run)
```

---

**cbs_get_toc**

Retrieve a data.frame with requested cbs tables

**Description**

cbs_get_toc by default a list of all tables and all columns will be retrieved. You can restrict the query by supplying multiple filter statements or by specifying the columns that should be returned.

**Usage**

```r
cbs_get_toc(
  ..., 
  convert_dates = TRUE, 
  select = NULL, 
  verbose = FALSE, 
  cache = TRUE, 
  base_url = getOption("cbsodataR.base_url", BASE_URL), 
  include_ID = FALSE
)
```

**Arguments**

- `...` filter statement to select rows, e.g. `Language="nl"`
- `convert_dates` convert the columns with date-time information into DateTime (default `TRUE`)
- `select` character columns to be returned, by default all columns will be returned.
- `verbose` logical prints the calls to the webservice
- `cache` logical should the result be cached?
- `base_url` optionally specify a different server. Useful for third party data services implementing the same protocol.
- `include_ID` logical column needed by OData but with no current use.
Value
data.frame with identifiers, titles and descriptions of tables

Note
cbs_get_toc will cache results, so subsequent calls will be much faster.

Examples
## Not run:
# get list of english tables
tables_en <- cbs_get_toc(Language="en")

# get list of dutch tables
tables_nl <- cbs_get_toc(Language="nl")
View(tables_nl)

## End(Not run)

cbs_join_sf_with_data  Create a map with data from cbsodataR

Description
Utility function to create an sf map object with data from cbsodataR.

Usage
cbs_join_sf_with_data(region, year, x, verbose = FALSE)

Arguments
region  character name of region
year  integer year of a region
x  data retrieved with cbs_get_data()
verbose  if TRUE the method is verbose

Details
The function is a simple wrapper around cbs_add_statcode_column() and cbs_get_sf(). Please note that the resulting sf::st_sf() dataset has the same number of rows as the requested map object, as requested by cbs_get_sf(), i.e. not the same rows as x. It’s the users responsibility to match the correct map to the selection of the data.

See Also
Other cartographic map: cbs_add_statcode_column(), cbs_get_maps(), cbs_get_sf()
Examples

```r
if (interactive()){

  # retrieve maps
  cbs_maps <- cbs_get_maps()
  cbs_maps |> head(4)

  gemeente_map <- cbs_get_sf("gemeente", 2023, verbose=TRUE)

  # sf object
  gemeente_map

  # plot the statcodes (included in the map)
  plot(gemeente_map, max.plot = 1)

  # now connect with some data
  labor <- cbs_get_data("85268NED"
      , Perioden = "2022JJ00" # only 2022
      , RegioS = has_substring("PV") # only province
      , verbose = TRUE
  )

  # most conveniently
  provincie_2022_with_data <- cbs_join_sf_with_data("provincie", 2022, labor)

  # better plotting options are ggplot2 or tmap,
  # but keeping dependencies low...
  provincie_2022_with_data |> subset(select = Werkloosheidspercentage_13) |> plot( border ="#FFFFFF99", main="unemployment rate")

  ## but of course this can also be done by hand:
  labor <- labor |>
    cbs_add_statcode_column() # add column to connect with map

  provincie_2022 <- cbs_get_sf("provincie", 2022)

  # this is a left_join(provincie_2022, labor, by = "statcode")
  provincie_2022_data <-
    within(provincie_2022, {
      unemployment_rate <- labor$Werkloosheidspercentage_13[match(statcode, labor$statcode)]
    })

  # better plotting options are ggplot2 or tmap,
  # but keeping dependencies low...
  plot( provincie_2022_data[,c("unemployment_rate")]
      , border ="#FFFFFF99"
      , nbreaks = 12
  )
}
```
cbs_search

Find tables containing search words

Description

Find tables containing search words.

Usage

cbs_search(
  query, 
  catalog = "CBS", 
  language = "nl", 
  format = c("datasets", "docs", "raw"), 
  verbose = FALSE, 
  ...
)

Arguments

query character with the words to search for.
catalog the subset in which the table is to be found, see cbs_get_catalogs(), set to NULL to query all catalogs.
language should the "nl" (Dutch) or "en" (English) search index be used.
format format in which the result should be returned, see details
verbose logical should the communication with the server be shown?
...

Details

The format can be either:

- datasets: the same format as cbs_get_datasets(), with an extra score column.
- docs: the table results from the solr query,
- raw: the complete results from the solr query.

Examples

if (interactive()){
  # search for tables containing the word birth

ds_en <- cbs_search("Birth", language="en")
ds_en[1:3, c("Identifier", "ShortTitle")]

  # or in Dutch
download_data-deprecated

\small

\texttt{ds\_nl <- cbs\_search(c("geboorte"), language="nl")}
\texttt{ds\_nl[1:3, c("Identifier", "ShortTitle")]

# Search in an other catalog
\texttt{ds\_rivm <- cbs\_search(c("geboorte"), catalog = "RIVM", language="nl")}
\texttt{ds\_rivm[1:3, c("Identifier", "ShortTitle")]

# search in all catalogs
\texttt{ds\_all <- cbs\_search(c("geboorte"), catalog = NULL, language="nl")}

# docs
\texttt{docs <- cbs\_search(c("geboorte,sterfte"), language="nl", format="docs")}
\texttt{names(docs)
\texttt{docs[1:2,]}

# raw
\texttt{raw\_res <- cbs\_search(c("geboorte,sterfte"), language="nl", format="raw")
\texttt{raw\_res
}

}\n
\vspace{1em}
\textbf{download_data-deprecated}

\emph{Gets all data from a cbs table.}

\section*{Description}

This method is deprecated in favor of \texttt{cbs\_download\_data()}. 

\section*{Usage}

\begin{verbatim}
download_data(id, path = file.path(id, "data.csv"), ...
select = NULL, typed = FALSE, verbose = TRUE,
base_url = getOption("cbsodataR.base_url", BASE_URL)
)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
  \item \texttt{id} \hspace{1cm} of cbs open data table
  \item \texttt{path} \hspace{1cm} of data file, defaults to "id/data.csv"
  \item \ldots \hspace{1cm} optional filter statements to select rows of the data,
  \item \texttt{select} \hspace{1cm} optional names of columns to be returned.
  \item \texttt{typed} \hspace{1cm} Should the data automatically be converted into integer and numeric?
\end{itemize}
download_meta-deprecated

Dumps the meta data into a directory

Description

This method is deprecated in favor of `cbs_download_meta()`.

Usage

```r
download_meta(
  id,
  dir = id,
  ...,
  verbose = FALSE,
  cache = FALSE,
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)
```

Arguments

- `id`  
  Id of CBS open data table (see `cbs_get_toc()`)

- `dir`  
  Directory in which data should be stored. By default it creates a sub directory with the name of the id

- `...`  
  not used

- `verbose`  
  Print extra messages what is happening.

- `cache`  
  Should meta data be cached?

- `base_url`  
  optionally allow to specify a different server. Useful for third party data services implementing the same protocol.

Value

- meta data object
See Also

Other meta data: `cbs_add_date_column()`, `cbs_add_label_columns()`, `cbs_get_meta()`
Other download: `cbs_download_data()`, `cbs_download_table()`

download_table-deprecated

*Download a table from statistics Netherlands*

download_table-deprecated

Description

This method is deprecated in favor of `cbs_download_table()`.

Usage

```r
download_table(
  id,                  
  ...,                
  dir = id,           
  cache = FALSE,      
  verbose = TRUE,     
  typed = FALSE,      
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)
```

Arguments

- **id**: Identifier of CBS table (can be retrieved from `cbs_get_toc()`)
- **...**: Parameters passed on to `cbs_download_data()`
- **dir**: Directory where table should be downloaded
- **cache**: If metadata is cached use that, otherwise download meta data
- **verbose**: Print extra messages what is happening.
- **typed**: Should the data automatically be converted into integer and numeric?
- **base_url**: optionally specify a different server. Useful for third party data services implementing the same protocol.

Details

`cbs_download_table` retrieves all raw meta data and data and stores these as csv files in the directory specified by `dir`. It is possible to add a filter. A filter is specified with `<column_name> = <values>` in which `<values>` is a character vector. Rows with values that are not part of the character vector are not returned.

Value

meta data object of id `cbs_get_meta()`.
Description

Detects for codes in a column. `eq` filters the data set at CBS: rows that have a code that is not in `x` are filtered out.

Usage

```r
eq(x, column = NULL, allowed = NULL)
```

Arguments

- `x`: exact code(s) to be matched in column
- `column`: name of column.
- `allowed`: character with allowed values. If supplied it will check if `x` is a code in `allowed`.

Value

query object

See Also

Other query: `cbs_get_data()`, `has_substring()`

Examples

```r
## Not run:
cbs_get_data(id = "7196ENG", Periods = "2000MM03", CPI = "000000")
# useful substrings:
```
get_data-deprecated

## Periods: "JJ": years, "KW": quarters, "MM": months
## Regions: "NL", "PV": provinces, "GM": municipalities

cbs_get_data(id = "7196ENG" # table id
, Periods = has_substring("JJ") # all years
, CPI = "000000" # Category code for total
)

cbs_get_data(id = "7196ENG" # table id
, Periods = c("2000MM03","2001MM12") # March 2000 and Dec 2001
, CPI = "000000" # Category code for total
)

# combine either this
cbs_get_data(id = "7196ENG" # table id
, Periods = has_substring("JJ") | "2000MM01" # all years and Jan 2001
, CPI = "000000" # Category code for total
)

# or this: note the "eq" function
cbs_get_data(id = "7196ENG" # table id
, Periods = eq("2000MM01") | has_substring("JJ") # Jan 2000 and all years
, CPI = "000000" # Category code for total
)

## End(Not run)

---

get_data-deprecated  
Get data from Statistics Netherlands (CBS)

Description

This method is deprecated in favor of cbs_get_data()

Usage

get_data(id,
...,
recode = TRUE,
use_column_title = recode,
dir = tempdir(),
base_url = getOption("cbsodataR.base_url", BASE_URL)
)

Arguments

id  Identifier of table, can be found in cbs_get_datasets()
...  optional filter statements, see details.
recode recodes all codes in the code columns with their Title as found in the metadata

use_column_title not used.

dir Directory where the table should be downloaded. Defaults to temporary directory

base_url optionally specify a different server. Useful for third party data services implementing the same protocol.

Details

To reduce the download time, optionally the data can be filtered on category values: for large tables (> 100k records) this is a wise thing to do.

The filter is specified with (see examples below):

- `<column_name> = <values>` in which `<values>` is a character vector. Rows with values that are not part of the character vector are not returned. **Note that the values have to be values from the $Key column of the corresponding meta data. These may contain trailing spaces.**

- `<column_name> = has_substring(x)` in which x is a character vector. Rows with values that do not have a substring that is in x are not returned. Useful substrings are "JJ", "KW", "MM" for Periods (years, quarters, months) and "PV", "CR" and "GM" for Regions (provinces, corops, municipalities).

- `<column_name> = eq(<values>) | has_substring(x)`, which combines the two statements above.

By default the columns will be converted to their type (typed=TRUE). CBS uses multiple types of missing (unknown, suppressed, not measured, missing): users wanting all these nuances can use typed=FALSE which results in character columns.

Value

data.frame with the requested data. Note that a csv copy of the data is stored in dir.

Copyright use

The content of CBS opendata is subject to Creative Commons Attribution (CC BY 4.0). This means that the re-use of the content is permitted, provided Statistics Netherlands is cited as the source. For more information see: [https://www.cbs.nl/en-gb/about-us/website/copyright](https://www.cbs.nl/en-gb/about-us/website/copyright)

Note

All data are downloaded using `cbs_download_table()`

See Also

`cbs_get_meta()`, `cbs_download_data()`

Other data retrieval: `cbs_add_date_column()`, `cbs_add_label_columns()`, `cbs_download_data()`, `cbs_extract_table_id()`, `cbs_get_data_from_link()`

Other query: `eq()`, `has_substring()`
Examples

## Not run:
cbs_get_data(id = "7196ENG" # table id,
    Periods = "2000MM03" # March 2000
    )

# useful substrings:
## Periods: "JJ": years, "KW": quarters, "MM", months
## Regions: "NL", "PV": provinces, "GM": municipalities

cbs_get_data(id = "7196ENG" # table id,
    Periods = has_substring("JJ") # all years
    )

cbs_get_data(id = "7196ENG" # table id,
    Periods = c("2000MM03","2001MM12") # March 2000 and Dec 2001
    )

# combine either this

cbs_get_data(id = "7196ENG" # table id,
    Periods = has_substring("JJ") | "2000MM01" # all years and Jan 2001
    )

# or this: note the "eq" function

cbs_get_data(id = "7196ENG" # table id,
    Periods = eq("2000MM01") | has_substring("JJ") # Jan 2000 and all years
    )

## End(Not run)

get_meta-deprecated  Get meta data from table

Description

This method is deprecated in favor of cbs_get_meta()

Usage

get_meta(id,
    verbose = TRUE,
    cache = FALSE,
    base_url = getOption("cbsodataR.base_url", BASE_URL)
)
Arguments

id          internal id of CBS table, can be retrieved with \texttt{cbs_get_datasets()}
verbose     Print extra messages what is happening.
cache       should the result be cached?
base_url    optionally specify a different server. Useful for third party data services implementing the same protocol.

Details

The meta data of a CBS table is determined by the web api of Statistics Netherlands. \texttt{cbsodataR} stays close to this API. Each \texttt{cbsodataR} object has the following metadata items, which are all \texttt{data.frame}s:

- $\$TableInfos$: \texttt{data.frame} with the descriptive publication metadata of the table, such as \texttt{Title}, \texttt{Description}, \texttt{Summary} etc.
- $\$DataProperties$: \texttt{data.frame} with the \texttt{Title}, \texttt{Description}, \texttt{Unit} etc. of each column in the dataset that is downloaded with \texttt{cbs_get_data()}.
- $\$CategoryGroups$: hierarchical groupings of the code columns.
- $\$<code column>$: for each code column a \texttt{data.frame} with the \texttt{Title}, \texttt{Key}, \texttt{Description} etc. of each code / category in that column. e.g. \texttt{Perioden} for time codes \texttt{c("2019JJ00","2018JJ00")}.

Value

cbs\_table object containing several \texttt{data.frame}s with meta data (see details)

See Also

Other meta data: \texttt{cbs_add_date_column()}, \texttt{cbs_add_label_columns()}, \texttt{cbs_download_meta()}

---

\texttt{get_meta_from_dir} \hspace{1cm} \textit{Load meta data from a downloaded table}

Description

Load meta data from a downloaded table

Usage

\texttt{get_meta_from_dir(dir)}

Arguments

dir         Directory where data was downloaded

Value

cbs\_table object with meta data
get_tables_themes

Get a the list of tables connected to themes

Description
Get a the list of tables connected to themes

Usage
get_tables_themes(
  ..., 
  select = NULL,
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)

Arguments

... Use this to add a filter to the query e.g. get_tables_themes(ID=10).
select character vector with names of wanted properties. default is all
base_url optionally specify a different server. Useful for third party data services implementing the same protocol.

Value
A data.frame with various properties of SN/CBS themes.

get_table_list
Retrieve a data.frame with requested cbs tables

Description
This method is deprecated in favor of cbs_get_toc().

Usage
get_table_list(
  ..., 
  select = NULL,
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)

Arguments

... filter statement to select rows, e.g. Language="nl"
select character columns to be returned, by default all columns will be returned.
base_url optionally specify a different server. Useful for third party data services implementing the same protocol.
get_themes

Value
data.frame with identifiers, titles and descriptions of tables

Examples
## Not run:

# get list of english tables
tables_en <- get_table_list(Language="en")

# get list of dutch tables
tables_nl <- get_table_list(Language="nl")
View(tables_nl)
## End(Not run)

get_themes

Get list of all cbs thematic entries.

Description
Returns a list of all cbs themes.

Usage
get_themes(
  ...,
  select = NULL,
  verbose = TRUE,
  cache = FALSE,
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)

Arguments
... Use this to add a filter to the query e.g. get_themes(ID=10).
select character vector with names of wanted properties. default is all
verbose Print extra messages what is happening.
cache Should the result be cached?
base_url optionally specify a different server. Useful for third party data services imple-
menting the same protocal.

Value
A data.frame with various properties of SN/CBS themes.
The filter is specified with <column_name> = <values> in which <values> is a character vector.
Rows with values that are not part of the character vector are not returned.
Examples

```r
## Not run:
# get list of all themes
get_themes()

# get list of all dutch themes from the Catalog "CBS"
get_themes(Language="nl", Catalog="CBS")

## End(Not run)
```

### has_substring

**Detect substring in column column**

Description

Detects a substring in a column. `has_substring` filters the dataset at CBS: rows that have a code that does not contain (one of) `x` are filtered out.

Usage

```r
has_substring(x, column = NULL, allowed = NULL)
```

Arguments

- `x` substring to be detected in column
- `column` column name
- `allowed` character with allowed values. If supplied it will check if `x` is a code in `allowed`.

See Also

Other query: `cbs_get_data()`, `eq()`

Examples

```r
## Not run:
cbs_get_data(id = "7196ENG" # table id
, Periods = "2000MM03" # March 2000
, CPI = "000000" # Category code for total )

# useful substrings:
## Periods: "JJ": years, "KW": quarters, "MM", months
## Regions: "NL", "PV": provinces, "GM": municipalities

cbs_get_data(id = "7196ENG" # table id
, Periods = has_substring("JJ") # all years
, CPI = "000000" # Category code for total
```
resolve_deeplink

resolve a deeplink created in the opendata portal

Description

resolve a deeplink created in the opendata portal

Usage

resolve_deeplink(
  deeplink,
  ..., 
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)

Arguments

deeplink url to the deeplink in the opendataportal
...
base_url optionally specify a different server. Useful for third party data services implementing the same protocol.

Value

information object with table id, select, filter and query statement.
Index

* cartographic map
  cbs_add_statcode_column, 6
  cbs_get_maps, 17
  cbs_get_sf, 20
  cbs_join_sf_with_data, 24

* data retrieval
  cbs_add_date_column, 4
  cbs_add_label_columns, 5
  cbs_download_data, 8
  cbs_extract_table_id, 11
  cbs_get_data, 12
  cbs_get_data_from_link, 16

* download
  cbs_download_data, 8
  cbs_download_data(), 10

* meta data
  cbs_add_date_column, 4
  cbs_add_label_columns, 5
  cbs_download_data(), 9
  cbs_get_meta, 18

* query
  cbs_get_data, 12
  eq, 30
  has_substring, 37
  cache_clear, 4
  cbs_add_date_column, 4, 5, 9–11, 14, 16, 19, 28, 29, 32, 34
  cbs_add_date_column(), 3
  cbs_add_label_columns, 4, 5, 9–11, 14, 16, 19, 28, 29, 32, 34
  cbs_add_label_columns(), 3
  cbs_add_statcode_column, 6, 17, 20, 24
  cbs_add_statcode_column(), 24
  cbs_default_selection, 8
  cbs_download_data, 4, 5, 8, 10, 11, 14, 16, 29, 30, 32
  cbs_download_data(), 10, 14, 27, 29, 32
  cbs_download_meta, 4, 5, 9, 11, 19, 28, 30, 34
  cbs_download_meta(), 28
  cbs_download_table, 9, 10, 10, 28, 29
  cbs_download_table(), 3, 14, 29, 32
  cbs_extract_table_id, 4, 5, 9, 11, 14, 16, 28, 32
  cbs_get_catalogs, 12
  cbs_get_catalogs(), 2, 15, 26
  cbs_get_data, 4, 5, 9, 11, 12, 16, 28, 30, 37
  cbs_get_data(), 2–6, 19, 24, 31, 34
  cbs_get_data_from_link, 4, 5, 9, 11, 14, 16, 28, 32
  cbs_getDatasets, 15
  cbs_getDatasets(), 2–6, 10, 12, 13, 18, 26, 31, 34
  cbs_get_maps, 6, 17, 20, 24
  cbs_get_maps(), 2, 3
  cbs_get_meta, 4, 5, 10, 18, 29
  cbs_get_meta(), 3, 11, 14, 29, 32, 33
  cbs_get_meta_from_dir, 19
  cbs_get_sf, 5, 17, 20, 24
  cbs_get_sf(), 3, 6, 24
  cbs_get_tables_themes, 21
  cbs_get_themes, 22
  cbs_get_toc, 23
  cbs_get_toc(), 3, 9, 10, 28, 29, 35
  cbs_join_sf_with_data, 6, 17, 20, 24
  cbs_join_sf_with_data(), 3
  cbs_search, 26
  cbs_search(), 3
  cbsodataR (cbsodataR-package), 2
  cbsodataR-package, 2
  Date(), 4
  download_data (download_data-deprecated), 27
  download_data-deprecated, 27
  download_meta (download_meta-deprecated), 28
download_meta-deprecated, 28
download_table
  (download_table-deprecated), 29
download_table-deprecated, 29

eq, 14, 30, 32, 37

get_data (get_data-deprecated), 31
get_data-deprecated, 31
get_meta (get_meta-deprecated), 33
get_meta-deprecated, 33
get_meta_from_dir, 34
get_table_list, 35
get_tables_themes, 35
get_themes, 36

has_substring, 14, 30, 32, 37

resolve_deeplink, 38

sf::st_sf(), 20, 24