Package ‘cbsodataR’

October 12, 2022

Type Package
Title Statistics Netherlands (CBS) Open Data API Client
Version 0.5.1
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)
VignetteBuilder knitr
RoxygenNote 7.1.1
NeedsCompilation no
Author Edwin de Jonge [aut, cre], Sara Houweling [ctb]
Maintainer Edwin de Jonge <edwindjonge@gmail.com>
Repository CRAN
Date/Publication 2020-09-14 12:40:02 UTC

R topics documented:
cbsodataR-package ................................................. 2
cache_clear .................................................. 3
cbs_add_date_column ....................................... 4
cbs_add_label_columns .................................... 5
cbs_default_selection ...................................... 6
cbs_download_data .......................................... 6
cbs_download_meta .......................................... 7
cbsodataR-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"). The functions cbs_get_toc() and cbs_get_data() should get you going.

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.

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)

---

cache_clear clears the cache

cache_clear()

description clears the cache

Usage

```r
cache_clear()
```
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. 2018JJ00, 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

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 2018K01 -> 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

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

## 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

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 accompagnied 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_download_data

Examples

## Not run:

```r
# get data for main (000000) Consumer Price Index (7196ENG) for March 2000,
x <- cbs_get_data(id = "7196ENG",
                  Periods = "2000MM03" # March 2000
                  CPI = "000000" # main price index
)
cbs_add_label_columns(x)

## End(Not run)
```

cbs_default_selection extract the default selection from a cbsodata meta object

Description

extract the default selection from a cbsodata meta object

Usage

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

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("cbsdodataR.base_url", BASE_URL)
)
```
cbs_download_meta

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Id of CBS open data table (see cbs_get_toc())</td>
</tr>
<tr>
<td>dir</td>
<td>Directory in which data should be stored. By default it creates a sub directory with the name of the id</td>
</tr>
<tr>
<td>catalog</td>
<td>catalog id, can be retrieved with cbs_get_datasets()</td>
</tr>
<tr>
<td>...</td>
<td>optional filter statements to select rows of the data,</td>
</tr>
<tr>
<td>select</td>
<td>optional names of columns to be returned.</td>
</tr>
<tr>
<td>typed</td>
<td>Should the data automatically be converted into integer and numeric?</td>
</tr>
<tr>
<td>verbose</td>
<td>show the underlying downloading of the data</td>
</tr>
<tr>
<td>show_progress</td>
<td>show a progress bar while downloading.</td>
</tr>
<tr>
<td>base_url</td>
<td>optionally specify a different server. Useful for third party data services implementing the same protocol.</td>
</tr>
</tbody>
</table>

See Also

Other download: cbs_download_meta(), cbs_download_table()
Other data retrieval: cbs_add_date_column(), cbs_add_label_columns(), cbs_extract_table_id(), cbs_get_data_from_link(), cbs_get_data()

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Id of CBS open data table (see cbs_get_toc())</td>
</tr>
<tr>
<td>dir</td>
<td>Directory in which data should be stored. By default it creates a sub directory with the name of the id</td>
</tr>
<tr>
<td>catalog</td>
<td>catalog id, can be retrieved with cbs_get_datasets()</td>
</tr>
</tbody>
</table>
cbs_download_table

... 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()

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

Usage
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 <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().

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

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**

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

**Description**

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

**Usage**

```r
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**

```r
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**

*Get data from Statistics Netherlands (CBS)*

**Description**

Retrieves data from a table of Statistics Netherlands. A list of available tables can be retrieved with `cbs_get_toc()`. Use the `Identifier` column of `cbs_get_toc` as `id` in `cbs_get_data` and `cbs_get_meta`.

**Usage**

```r
cbs_get_data(
  id,
  ...,
  catalog = "CBS",
  select = NULL,
  typed = TRUE,
)```
cbs_get_data

    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_toc()
... optional filter statements, see details.
catalog catalog id, can be retrieved with cbs_get_datasets()
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, 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.
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

```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
 )

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)
**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 de datasets in the "AZW" catalog
  ds_azw <- cbs_get_datasets(catalog = "AZW")
  ```
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

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>link</td>
<td>url/hyperlink to opendata table made with the StatLine App</td>
</tr>
<tr>
<td>message</td>
<td>logical Should the query be printed (default TRUE)</td>
</tr>
<tr>
<td>...</td>
<td>passed on to cbs_get_data</td>
</tr>
<tr>
<td>base_url</td>
<td>optionally specify a different server. Useful for third party data services implementing the same protocol.</td>
</tr>
</tbody>
</table>

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_data()
**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**
```r
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()`  
- `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. cbsodataR stays close to this API. Each cbsodataR object has the following metadata items, which are all data.frames:

- `$TableInfos`: data.frame with the descriptive publication metadata of the table, such as Title, Description, Summary etc.
- `$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.frames 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
cbs_get_meta_from_dir(dir)
```

**Arguments**

- `dir`: Directory where data was downloaded

**Value**

`cbs_table` object with meta data

---

### cbs_get_tables_themes

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

**Description**

Get a the list of tables connected to themes

**Usage**

```r
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

## 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

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")
cbs_search

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

## End(Not run)

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?
... not used

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

```r
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
  ds_nl <- cbs_search(c("geboorte"), language="nl")
  ds_nl[1:3, c("Identifier", "ShortTitle")]

  # Search in an other catalog
  ds_rivm <- cbs_search(c("geboorte"), catalog = "RIVM", language="nl")
  ds_rivm[1:3, c("Identifier", "ShortTitle")]

  # search in all catalogs
  ds_all <- cbs_search(c("geboorte"), catalog = NULL, language="nl")

  # docs
  docs <- cbs_search(c("geboorte,sterfte"), language="nl", format="docs")
  names(docs)
  docs[1:2,]

  #raw
  raw_res <- cbs_search(c("geboorte,sterfte"), language="nl", format="raw")
  raw_res
}
```

---

download_data-deprecated

*Gets all data from a cbs table.*

Description

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

Usage

```r
download_data(
id,
  path = file.path(id, "data.csv"),
  ..., 
  select = NULL,
  typed = FALSE,
  verbose = TRUE,
  base_url = getOption("cbsodataR.base_url", BASE_URL)
)
```
Arguments

- **id**: Id of CBS open data table
- **path**: of data file, defaults to "id/data.csv"
- **...**: optional filter statements to select rows of the data
- **select**: optional names of columns to be returned.
- **typed**: Should the data automatically be converted into integer and numeric?
- **verbose**: show the underlying downloading of the data
- **base_url**: optionally specify a different server. Useful for third party data services implementing the same protocol.

See Also

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

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

---

download_meta-deprecated

*Dumps the meta data into a directory*

Description

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

Usage

download_meta(
  id,
  dir = id,
  ...,
  verbose = FALSE,
  cache = FALSE,
  base_url = getOption("cbsdataR.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*

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()`.

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)
```

---

### eq

**Detect codes in a column**

**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**

`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" # 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
, 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

```r
get_data(
  id,
  ..., 
  recode = TRUE,
  use_column_title = recode,
```
get_data-deprecated

```r
dir = tempdir(),
base_url = getOption("cbsodataR.base_url", BASE_URL)
```

### Arguments

- **id**: Identifier of table, can be found in `cbs_get_toc()`
- ... 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 metadata. 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 "JI", "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.

### 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_download_data_from_link()`

Other query: `eq()`, `has_substring()`
get_meta-deprecated

Get meta data from table

Description

This method is deprecated in favor of `cbs_get_meta()`

Usage

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

Arguments

- **id**: internal id of CBS table, can be retrieved with `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. cbsodataR stays close to this API. Each cbsodataR object has the following metadata items, which are all data.frames:

- `$TableInfos`: data.frame with the descriptive publication metadata of the table, such as Title, Description, Summary etc.
- `$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("2019J100","2018J100")`.

Value

cbs_table object containing several data.frames with meta data (see details)

See Also

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

Description

Load meta data from a downloaded table

Usage

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-

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.
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**

`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 = has_substring("JJ") # all years
, CPI = "000000" # Category code for total
)
```
### Description

`resolve_deeplink` resolve a deeplink created in the opendata portal

### Usage

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

### Arguments

- `deeplink`: url to the deeplink in the opendataportal
- `...`: used in the query
- `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

* data retrieval
  - cbs_add_date_column, 4
  - cbs_add_label_columns, 5
  - cbs_download_data, 6
  - cbs_extract_table_id, 9
  - cbs_get_data, 10
  - cbs_get_data_from_link, 14
  - cbs_get_catalogs(), 2, 13, 19
  - cbs_get_data, 4, 5, 7, 9, 10, 14, 21, 23, 30
  - cbs_get_data(), 2–5, 15, 24, 27
  - cbs_get_data_from_link, 4, 5, 7, 9, 12, 14, 21, 25
  - cbs_get_datasets, 13
  - cbs_get_datasets(), 2, 7, 8, 11, 15, 19, 27
  - cbs_get_meta, 4, 5, 8, 15, 22
  - cbs_get_meta(), 3, 9, 12, 23, 25, 26
  - cbs_get_meta_from_dir, 16
  - cbs_get_tables_themes, 16
  - cbs_get_themes, 17
  - cbs_get_toc, 18
  - cbs_get_toc(), 2, 7, 8, 10, 11, 21, 22, 25, 28
  - cbs_search, 19
  - cbs_search(), 2
  - cbsodataR (cbsodataR-package), 2
  - cbsodataR-package, 2
  - Date(), 4
  - download_data
    - (download_data-deprecated), 20
  - download_data-deprecated, 20
  - download_meta
    - (download_meta-deprecated), 21
  - download_meta-deprecated, 21
  - download_table
    - (download_table-deprecated), 22
  - download_table-deprecated, 22
  - eq, 12, 23, 25, 30
  - get_data (get_data-deprecated), 24
  - get_data-deprecated, 24
  - get_meta (get_meta-deprecated), 26
  - get_meta-deprecated, 26
  - get_meta_from_dir, 27
  - get_table_list, 28
  - get_tables_themes, 28
  - get_themes, 29

* download
  - cbs_download_data, 6
  - cbs_download_meta, 7
  - cbs_download_table, 8

* meta data
  - cbs_add_date_column, 4
  - cbs_add_label_columns, 5
  - cbs_download_meta, 7
  - cbs_get_meta, 15

* query
  - cbs_get_data, 10
  - eq, 23
  - has_substring, 30
  - cache_clear, 3
  - cbs_add_date_column, 4, 5, 7–9, 12, 14, 15, 21, 22, 25, 27
  - cbs_add_date_column(), 3
  - cbs_add_label_columns, 4, 5, 7–9, 12, 14, 15, 21, 22, 25, 27
  - cbs_add_label_columns(), 3
  - cbs_default_selection, 6
  - cbs_download_data, 4, 5, 6, 8, 9, 12, 14, 22, 23, 25
  - cbs_download_data(), 8, 12, 20, 22, 25
  - cbs_download_meta, 4, 5, 7, 9, 15, 21, 23, 27
  - cbs_download_meta(), 21
  - cbs_download_table, 7, 8, 21, 22
  - cbs_download_table(), 3, 12, 22
  - cbs_extract_table_id, 4, 5, 7, 9, 12, 14, 21, 25
  - cbs_get_catalogs, 10
INDEX

has_substring, 12, 23, 25, 30

resolve_deeplink, 31