Package ‘cancensus’

June 8, 2021

Type Package

Title Access, Retrieve, and Work with Canadian Census Data and Geography

Version 0.4.3

Description Integrated, convenient, and uniform access to Canadian Census data and geography retrieved using the ‘CensusMapper’ API. This package produces analysis-ready tidy data frames and spatial data in multiple formats, as well as convenience functions for working with Census variables, variable hierarchies, and region selection. API keys are freely available with free registration at <https://censusmapper.ca/api>. Census data and boundary geometries are reproduced and distributed on an “as is” basis with the permission of Statistics Canada (Statistics Canada 2001; 2006; 2011; 2016).

License MIT + file LICENSE

Encoding UTF-8

LazyData yes

ByteCompile yes

NeedsCompilation no

Imports digest (>= 0.1), dplyr (>= 0.7), httr (>= 1.0.0), jsonlite (>= 1.0), rlang

RoxygenNote 7.1.1

Suggests knitr, ggplot2, leaflet, mapdeck, rmarkdown, readr, rgdal, rgeos, scales, sp, sf, geojsonsf, tidyr, lwgeom

VignetteBuilder knitr

URL https://github.com/mountainMath/cancensus,
https://mountainmath.github.io/cancensus/,
https://censusmapper.ca/api

BugReports https://github.com/mountainMath/cancensus/issues

Depends R (>= 2.10)
as_census_region_list

Convert a (suitably filtered) data frame from list_census_regions to a list suitable for passing to get_census.

Usage

as_census_region_list(tbl)
census_vectors

Arguments

tbl A data frame, suitably filtered, as returned by `list_census_regions`.

Examples

```r
## Not run:
library(dplyr, warn.conflicts = FALSE)
# Query the CensusMapper API for the total occupied dwellings
# of 20 random Census Subdivisions, in Census 2016.
regions <- list_census_regions("CA16") %>%
  filter(level == "CSD") %>%
  sample_n(20) %>%
  as_census_region_list()
occupied <- get_census("CA16", regions = regions,
  vectors = c("v_CA16_408"),
  level = "Regions")
## End(Not run)
```

census_vectors

Return Census variable names and labels as a tidy data frame (Deprecated)

Description

Return Census variable names and labels as a tidy data frame (Deprecated)

Usage

census_vectors(x)

Arguments

x A data frame, sp or sf object returned from `get_census` or similar.

Value

A data frame with a column `variable` containing the truncated variable name, and a column `label` describing it.

Examples

```r
## Not run:
# Query census data with truncated labels:
census_data <- get_census(dataset="CA16", regions=list(CMA="59933"),
  vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"),
  level='CSD', geo_format = "sf", labels="short")
```
child_census_vectors

List all child variables from vector hierarchies given either a list of Census variables returned by list_census_vectors, search_census_vectors, find_census_vectors, or a direct string reference to the vector code.

Description

List all child variables from vector hierarchies given either a list of Census variables returned by list_census_vectors, search_census_vectors, find_census_vectors, or a direct string reference to the vector code.

Usage

child_census_vectors(
  vector_list,
  leaves_only = FALSE,
  max_level = NA,
  keep_parent = FALSE
)

Arguments

- `vector_list`: the list of vectors to be used, either a character vector or a filtered tibble as returned from list_census_vectors.
- `leaves_only`: boolean flag to indicate if only final leaf vectors should be returned, i.e. terminal vectors that themselves do not have children.
- `max_level`: optional, maximum depth to look for child vectors. Default is NA and will return all child census vectors.
- `keep_parent`: optional, also return parent vector in list of results. Default is set to FALSE.

Examples

# Query parent vectors directly using vector identifier
class_census_vectors("v_CA16_2510")

## Not run:

# Example using multiple vectors coerced into a list
class_census_vectors(c("v_CA16_2510","v_CA16_2511","v_CA16_2512"))
# or, equivalently
selected_vectors <- c("v_CA16_2510","v_CA16_2511","v_CA16_2512")
child_census_vectors(selected_vectors)

# Example using dplyr and piped arguments
library(dplyr, warn.conflicts = FALSE)

list_census_vectors("CA16") %>%
  filter(vector == "v_CA16_2510") %>%
  child_census_vectors(TRUE)

# this will return the equivalent of c("v_CA16_2510", child_census_vectors("v_CA16_2510"))
list_census_vectors("CA16") %>%
  filter(vector == "v_CA16_2510") %>%
  child_census_vectors(TRUE, keep_parent = TRUE)

## End(Not run)

---

**CODES_TABLE**

A dataset with code table summaries for census data

---

**Description**

A dataset with code table summaries for census data

**Author(s)**

derived from StatCan definitions

**References**


---

**COV_SKYTRAIN_STATIONS**

A dataset City of Vancouver skytrain station locations

---

**Description**

A dataset City of Vancouver skytrain station locations

**Author(s)**

City of Vancouver Open Data

**References**

https://opendata.vancouver.ca/explore/dataset/rapid-transit-stations/information/
dataset_attribution  
*Get attribution for datasets*

**Description**
Get attribution for datasets

**Usage**
```r
dataset_attribution(datasets)
```

**Arguments**
- **datasets**  
  Vector of dataset identifiers

**Value**
Returns a string to be used as attribution for the given datasets.

**Examples**
```r
# Attribution string for the 2006 and 2016 census datasets
dataset_attribution(c("CA06","CA16"))
```

explore_census_regions  
*Interactively browse Census variables and regions on Censusmapper.ca in a new browser window*

**Description**
Finding the right Census variables or regions can be complicated. `explore_census_vectors(dataset)` and `explore_census_regions(dataset)` will open a new browser page or tab to an interactive Census variable and region exploration and selection tool on the Censusmapper.ca website. Interactive tools available for the CA16, CA11, CA06, and CA01 Census datasets and geographies.

**Usage**
```r
explore_census_regions(dataset = "CA16")
```

**Arguments**
- **dataset**  
  The dataset to query for available vectors, e.g. 'CA16'. Interactive tools available for the CA16, CA11, CA06, and CA01 Census datasets and geographies.
explore_census_vectors

Examples

### Not run:

```r
explore_census_vectors(dataset = "CA16")

explore_census_regions(dataset = "CA11")
```

### End(Not run)

---

explore_census_vectors  
*Interactively browse Census variables and regions on Censusmapper.ca in a new browser window*

---

Description

Finding the right Census variables or regions can be complicated. `explore_census_vectors(dataset)` and `explore_census_regions(dataset)` will open a new browser page or tab to an interactive Census variable and region exploration and selection tool on the Censusmapper.ca website. Interactive tools available for the CA16, CA11, CA06, and CA01 Census datasets and geographies.

Usage

```r
explore_census_vectors(dataset = "CA16")
```

Arguments

- **dataset**  
  The dataset to query for available vectors, e.g. `"CA16"`. Interactive tools available for the CA16, CA11, CA06, and CA01 Census datasets and geographies.

Examples

### Not run:

```r
explore_census_vectors(dataset = "CA16")

explore_census_regions(dataset = "CA11")
```

### End(Not run)
find_census_vectors  

*Query the CensusMapper API for vectors using exact, semantic, or keyword search*

**Description**

Query the available list of Census vectors based on their label and return details including vector code. Default search behaviour expects an exact match, but keyword or semantic searches can be used instead by setting `query_type='keyword'` or `query_type='semantic'` instead. Keyword search is useful when looking to explore Census vectors based on broad themes like "income" or "language". Keyword search separates the query into unigrams and returns Census vectors with matching words, ranked by incidence of matches. Semantic search is designed for more precise searches while allowing room for error for spelling or phrasing, as well as for finding closely related vector matches. Semantic search separates the query into n-grams and relies on string distance measurement using a generalized Levenshtein distance approach.

Some census vectors return population counts segmented by Female and Male populations, in addition to a total aggregate. By default, query matches will return matches for the Total aggregation, but can optionally return only the Female or Male aggregations by adding `type='female'` or `type='male'` as a parameter.

**Usage**

```r
find_census_vectors(query, dataset, type = "all", query_type = "exact", ...)
```

**Arguments**

- **query**
  - The term or phrase to search for e.g. 'Oji-cree'. Search queries are case insensitive.

- **dataset**
  - The dataset to query for available vectors, e.g. 'CA16'. To see a list of available datasets: `list_census_datasets()`

- **type**
  - One of 'all', 'total', 'male' or 'female'. If specified, only return aggregations of specified 'type'. By default, only the 'total' aggregation will be returned.

- **query_type**
  - One of exact, 'semantic' or 'keyword'. By default, assumes exact string matching, but the alternatives may be better options in some cases. See description section for more details on query types.

- **...**
  - Other arguments passed to internal functions.

**Examples**

```r
find_census_vectors('Oji-cree', dataset = 'CA16', type = 'total', query_type = 'exact')
find_census_vectors('commuting duration', dataset = 'CA11', type = 'female', query_type = 'keyword')
find_census_vectors('after tax income', dataset = 'CA16', type = 'total', query_type = 'semantic')
```
## Not run:
# This incorrect spelling will return a warning that no match was found,  
# but will suggest trying semantic or keyword search.  
find_census_vectors('Ojibwey', dataset = 'CA16', type = 'total')

# This will find near matches as well  
find_census_vectors('Ojibwey', dataset = 'CA16', type = 'total', query_type = "semantic")

find_census_vectors('commute duration', dataset = 'CA16', type = 'female', query_type = 'keyword')

find_census_vectors('commute duration', dataset = 'CA11', type = 'all', query_type = 'keyword')

find_census_vectors('ukrainian origin', dataset = 'CA16', type = 'total', query_type = 'keyword')

## End(Not run)

---

**get_census**  
Access to Canadian census data through the CensusMapper API

### Description

This function allows convenient access to Canadian census data and boundary files through the CensusMapper API. An API key is required to retrieve data.

### Usage

```r
get_census(
  dataset,
  regions,
  level = NA,
  vectors = c(),
  geo_format = NA,
  labels = "detailed",
  use_cache = TRUE,
  quiet = FALSE,
  api_key = Sys.getenv("CM_API_KEY")
)
```

### Arguments

- **dataset**: A CensusMapper dataset identifier.
- **regions**: A named list of census regions to retrieve. Names must be valid census aggregation levels.
- **level**: The census aggregation level to retrieve, defaults to "Regions". One of "Regions", "PR", "CMA", "CD", "CSD", "CT", "DA", "EA" (for 1996), or "DB" (for 2001-2016).
get_census

vectors An R vector containing the CensusMapper variable names of the census variables to download. If no vectors are specified only geographic data will get downloaded.

geo_format By default is set to NA and appends no geographic information. To include geographic information with census data, specify one of either "sf" to return an sf object (requires the sf package) or "sp" to return a SpatialPolygonsDataFrame-class object (requires the rgdal package).

labels Set to "detailed" by default, but truncated Census variable names can be selected by setting labels = "short". Use label_vectors(...) to return variable label information in detail.

use_cache If set to TRUE (the default) data will be read from the local cache if available.

quiet When TRUE, suppress messages and warnings.

api_key An API key for the CensusMapper API. Defaults to options() and then the CM_API_KEY environment variable.

Details

For help selecting regions and vectors, see list_census_regions and list_census_vectors, or check out the interactive selection tool at https://censusmapper.ca/api by calling explore_census_vectors()

Source

Census data and boundary geographies are reproduced and distributed on an "as is" basis with the permission of Statistics Canada (Statistics Canada 1996; 2001; 2006; 2011; 2016).

Examples

# Query the API for data on dwellings in Vancouver, at the census subdivision
# level:
## Not run:
census_data <- get_census(dataset="CA16", regions=list(CMA="59933"),
 vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"),
 level="CSD")

# Query the API for data on dwellings in Vancouver, at the census subdivision
# level, and return the associated geography files in \( \text{code}(sf) \) format:
census_data <- get_census(dataset="CA16", regions=list(CMA="59933"),
 vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"),
 level="CSD", geo_format = "sf")

# Make the same query, but this time drop descriptive vector names:
census_data <- get_census(dataset="CA16", regions=list(CMA="59933"),
 vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"),
 level="CSD", geo_format = "sf", labels="short")

# Get details for truncated vectors:
label_vectors(census_data)

## End(Not run)
get_intersecting_geometries

Get identifiers for census regions intersecting a geometry

Description

This function returns a list of regions that intersect a given geometry input. This list of regions can be used directly to query census when one is interested in census data for a particular geographic region that does not coincide with defined census geometries. The returned value can be used as the regions parameter in get_census to get corresponding census geographies and variables that cover the given geometry. Input spatial objects can be any sf or sfc class objects such as POINT, MULTIPOLYGON, or POLYGON.

This function comes with CensusMapper API limits

Usage

get_intersecting_geometries(
    dataset,                 # A CensusMapper dataset identifier.
    level,                   # The census aggregation level to retrieve. One of "Regions", "PR", "CMA", "CD", "CSD", "CT", "DA", "EA" (for 1996 census), or "DB" (for 2001-2016 censuses).
    geometry,               # A valid sf or sfc class object. As long as the geometry is valid, any projection is accepted. Objects will be reprojected as server-side intersections use lat/lon projection.
    simplified = FALSE,     # If TRUE will return a region list compatible with get_census, otherwise will return a character vector of matching region ids.
    use_cache = TRUE,       # If set to TRUE (the default) data will be read from the local cache if available.
    quiet = FALSE,          # When TRUE, suppress messages and warnings.
    api_key = Sys.getenv("CM_API_KEY")  # An API key for the CensusMapper API. Defaults to options() and then the CM_API_KEY environment variable.
)

Arguments

dataset:
level:
geometry:
simplified:
use_cache:
quiet:
api_key:

Source

Census data and boundary geographies are reproduced and distributed on an "as is" basis with the permission of Statistics Canada (Statistics Canada 1996; 2001; 2006; 2011; 2016).
Examples

## Not run:
# Example using a POINT-class object from a pair of lat/lon coordinates
point_geo <- sf::st_sfc(sf::st_point(c(-123.25149, 49.27026)), crs=4326)
regions <- get_intersecting_geometries(dataset = 'CA16', level = 'CT', geometry = point_geo)
census_data <- get_census(dataset='CA16', regions=regions,
  vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"),
  level='CT')

# Example using a POLYGON-class object from a bounding box with four coordinates
poly_geo <- sf::st_as_sfc(sf::st_bbox(c(ymin = 49.25, ymax = 49.30,
  xmin = -123.25, xmax = -123.30)), crs = 4326)
regions <- get_intersecting_geometries(dataset = 'CA16', level = 'CT', geometry = poly_geo)
census_data <- get_census(dataset='CA16', regions=regions,
  vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"), level='CT')

## End(Not run)

---

**label_vectors**  
*Return Census variable names and labels as a tidy data frame*

## Description

Return Census variable names and labels as a tidy data frame

## Usage

`label_vectors(x)`

## Arguments

- `x`  
  A data frame, sp or sf object returned from `get_census` or similar.

## Value

A data frame with a column `variable` containing the truncated variable name, and a column `label` describing it.

## Examples

## Not run:
# Query census data with truncated labels:
label_data <- get_census(dataset='CA16', regions=list(CMA="59933"),
  vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"),
  level='CSD', geo_format = "sf", labels="short")

# Get details for truncated vectors:
list_census_datasets

label_vectors(label_data)

## End(Not run)

list_census_datasets  Query the CensusMapper API for available datasets.

Description

Query the CensusMapper API for available datasets.

Usage

list_census_datasets(use_cache = TRUE, quiet = FALSE)

Arguments

use_cache  If set to TRUE (the default), data will be read from a temporary local cache for the duration of the R session, if available. If set to FALSE, query the API for the data, and refresh the temporary cache with the result.

quiet  When TRUE, suppress messages and warnings.

Value

Returns a data frame with a column dataset containing the code for the dataset, a column description describing it, a geo_dataset column identifying the geography dataset the data is based on, an attribution column with an attribution string, a reference column with a reference identifier, and a reference_url column with a link to reference materials.

Examples

# List available datasets in CensusMapper
list_census_datasets()

list_census_regions  Query the CensusMapper API for available regions in a given dataset.

Description

Query the CensusMapper API for available regions in a given dataset.

Usage

list_census_regions(dataset, use_cache = TRUE, quiet = FALSE)
Arguments

- **dataset**
  The dataset to query for available regions, e.g. "CA16".

- **use_cache**
  If set to TRUE (the default), data will be read from a local cache that is maintained for the duration of the R session, if available. If set to FALSE, query the API for the data, and refresh the local cache with the result.

- **quiet**
  When TRUE, suppress messages and warnings.

Value

Returns a data frame with the following columns:

- **region**
  The region identifier.
- **name**
  The name of that region.
- **level**
  The census aggregation level of that region.
- **pop**
  The population of that region.
- **municipal_status**
  Additional identifiers to distinguish the municipal status of census subdivisions.
- **CMA_UID**
  The identifier for the Census Metropolitan Area the region is in (if any).
- **CD_UID**
  The identifier for the Census District the region is in (if any).
- **PR_UID**
  The identifier for the Province the region is in (if unique).

Examples

```r
list_census_regions('CA16')
```

---

**list_census_vectors**

Query the CensusMapper API for available vectors for a given dataset.

Description

Query the CensusMapper API for available vectors for a given dataset.

Usage

```r
list_census_vectors(dataset, use_cache = TRUE, quiet = TRUE)
```

Arguments

- **dataset**
  The dataset to query for available vectors, e.g. "CA16".

- **use_cache**
  If set to TRUE (the default), data will be read from a local cache that is maintained for the duration of the R session, if available. If set to FALSE, query the API for the data, and refresh the local cache with the result.

- **quiet**
  When FALSE, shows messages and warnings. Set to TRUE by default.
Value

Returns a data frame detailing the available Census vectors (i.e. variables) for a given Census dataset. This data frame has columns `vector` containing the short code for the variable, `type` describing whether it's a female, male, or total aggregate, `label` indicating the name of the variable, `units` indicating whether the value represents a numeric integer, percentage, dollar figure, or ratio, `parent_vector` to show hierarchical relationship, `aggregation` indicating whether the value is additive or a transformation, and a column `details` with a detailed description of the variable generated by traversing all labels within its hierarchical structure.

Examples

```r
## Not run:
# List all vectors for a given Census dataset in CensusMapper
list_census_vectors('CA16')
## End(Not run)
```

### Description

List all parent variables from vector hierarchies given either a list of Census variables returned by `list_census_vectors`, `search_census_vectors`, `find_census_vectors`, or a direct string reference to the vector code.

### Usage

```r
parent_census_vectors(vector_list)
```

### Arguments

- **vector_list**: The list of vectors to be used, either a character vector or a filtered tibble as returned from `list_census_vectors`.

### Examples

```r
# Query parent vectors directly using vector identifier
parent_census_vectors("v_CA16_2519")
## Not run:
# Example using multiple vectors coerced into a list
parent_census_vectors(c("v_CA16_2519","v_CA16_2520","v_CA16_2521"))
# or, equivalently
```
search_census_regions

Query the CensusMapper API for regions with names matching a searchterm.

Description

Runs a query against the CensusMapper API to retrieve region data with names matching specific queries. Users can optionally specify the target geography level (e.g. `level = "CMA"`, `level = "CSD"`, etc.). Alternatively, calling `explore_census_vectors()` will launch the interactive region selection tool on the Censusmapper site in a new web page or tab.

Usage

```r
search_census_regions(searchterm, dataset, level = NA, ...)
```

Arguments

- `searchterm` The term to search for e.g. "Victoria". Search terms are case insensitive. If unable to find a given search term, this function will suggest the correct spelling to use when possible.
- `dataset` The dataset to query for available regions, e.g. "CA16".
- `level` One of `NA`, 'C', 'PR', 'CMA', 'CD', or 'CSD'. If specified, only return variables of specified `level`.
- `...` Further arguments passed on to `list_census_regions`.

Examples

```r
search_census_regions('Victoria', 'CA16')

## Not run:
# This will return a warning that no match was found, but will suggest similar named regions.
search_census_regions('Victoria', 'CA16')

# This will limit region results to only include CMA level regions
search_census_regions('Victoria', 'CA16', level = "CMA")

## End(Not run)
```
search_census_vectors  \textit{Query the CensusMapper API for vectors with descriptions matching a search term or phrase (deprecated)}

Description

Query the CensusMapper API for vectors with descriptions matching a search term or phrase (deprecated)

Usage

\begin{verbatim}
search_census_vectors(searchterm, dataset, type = NA, ...)
\end{verbatim}

Arguments

- \texttt{searchterm}: The term or phrase to search for e.g. "Ojibway". Search terms are case insensitive. If unable to find a given string, this function will suggest similarly named objects.
- \texttt{dataset}: The dataset to query for available vectors, e.g. "CA16".
- \texttt{type}: One of NA, 'Total', 'Male' or 'Female'. If specified, only return variables of specified 'type'.
- \texttt{...}: Further arguments passed on to \texttt{list_census_vectors}.

Examples

\begin{verbatim}
search_census_vectors('Ojibway', 'CA16')
## Not run:
# This will return a warning that no match was found, but will suggest similar terms.
search_census_vectors('Ojibwe', 'CA16', 'Total')

## End(Not run)
\end{verbatim}

set_api_key  \textit{Set Censusmapper API key}

Description

Cancensus requires a free Censusmapper API key to retrieve data. This function helps set the key for either the duration of the session (default) or permanently for use across sessions.

Usage

\begin{verbatim}
set_api_key(key, overwrite = FALSE, install = FALSE)
\end{verbatim}
### set_cache_path

Set persistent cancensus cache location

#### Description

Cancensus provides session caching for retrieved data to increase speeds and reduce API usage. This function will create a persistent cache across sessions.

#### Usage

```
set_cache_path(cache_path, overwrite = FALSE, install = FALSE)
```

#### Arguments

- `cache_path` : a local directory to use for saving cached data
- `overwrite` : Option to overwrite any existing Censusmapper keys already stored locally.
- `install` : Option to install permanently for use across sessions.

#### Examples

```r
## Not run:
set_cache_path("~/cancensus_cache")

# This will set the cache path permanently until overwritten again
set_cache_path("~/cancensus_cache", install = TRUE)

## End(Not run)
```
show_api_key

**Description**
View saved Censusmapper API key

**Usage**
```python
show_api_key()
```

---

show_cache_path

**Description**
View saved cache directory path

**Usage**
```python
show_cache_path()
```
Index

* data
  CODES_TABLE, 5
  COV_SKYTRAIN_STATIONS, 5

as_census_region_list, 2

census_vectors, 3
child_census_vectors, 4
CODES_TABLE, 5
COV_SKYTRAIN_STATIONS, 5

dataset_attribution, 6

explore_census_regions, 6
explore_census_vectors, 7

find_census_vectors, 8

get_census, 2, 9, 11
get_intersecting_geometries, 11

label_vectors, 12
list_census_datasets, 13
list_census_regions, 2, 3, 10, 13, 16
list_census_vectors, 10, 14, 17

parent_census_vectors, 15

search_census_regions, 16
search_census_vectors, 17
set_api_key, 17
set_cache_path, 18
sf, 10
show_api_key, 19
show_cache_path, 19

20