cancensus

Cancensus and CensusMapper

The **cancensus** package was developed to provide users with a way to access Canadian Census in a programmatic way following good tidy data practices. While the structure and data in **cancensus** is unique to Canadian Census data, this package is inspired in part by tidycensus, a package to interface with the US Census Bureau data APIs.

As Statistics Canada does not provide direct API access to Census data, **cancensus** retrieves Census data indirectly through the CensusMapper API. CensusMapper is a project by Jens von Bergmann, one of the authors of **cancensus**, to provide interactive geographic visualizations of Canadian Census data. CensusMapper databases store all publically available data from Statistics Canada for the 2006, 2011, and 2016 Censuses. Censusmapper data can be accessed via an API and **cancensus** is built to interface directly with it.

API Key

cancensus requires a valid CensusMapper API key to use. You can obtain a free API key by signing up for a CensusMapper account. CensusMapper API keys are free; however, API requests are limited in volume. For larger quotas, please get in touch with Jens directly.

To check your API key, just go to "Edit Profile" (in the top-right of the CensusMapper menu bar). Once you have your key, you can store it in your system environment so it is automatically used in API calls. To do so just enter `options(cancensus.api_key = "your_api_key")`.

Installing cancensus

```r
# install.packages("devtools")
devtools::install_github("mountainmath/cancensus")
library(cancensus)
options(cancensus.api_key = "your_api_key")
```

Accessing Census Data

cancensus provides three different functions for retrieving Census data: * `get_census` to retrieve Census data and geography as a spatial dataset * `get_census_data` to retrieve Census data only as a flat data frame * `get_census_geometry` to retrieve Census geography only as a collection of spatial polygons.

`get_census` takes as inputs a dataset parameter, a list of specified regions, a vector of Census variables, and a Census geography level. You can specify one of three options for spatial formats: `NA` to return data only, `sf` to return an sf-class data frame, or `sp` to return a SpatialPolygonsDataFrame object.

```r
# Returns a data frame with data only
census_data <- get_census(dataset='CA16', regions=list(CMA="59933"),
                          vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"),
                          level='CSD', use_cache = FALSE, geo_format = NA)
# Returns data and geography as an sf-class data frame
census_data <- get_census(dataset='CA16', regions=list(CMA="59933"),
                          vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"),
                          level='CSD', use_cache = FALSE, geo_format = 'sf')
# Returns a SpatialPolygonsDataFrame object with data and geogrpahy
census_data <- get_census(dataset='CA16', regions=list(CMA="59933"),
                          vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"),
                          level='CSD', use_cache = FALSE, geo_format = 'sp')
```

cancensus utilizes caching to increase speed, minimize API token usage, and to make data available offline. Downloaded data is hashed and stored locally so if a call is made to access the same data, **cancensus** will read the local version instead. To force **cancensus** to refresh the data, specify `use_cache = FALSE` as a parameter for `get_census`.

Additional parameters for advanced options can be viewed by running `?get_census`.

Census Datasets

cancensus can access Statistics Canada Census data for the 2006 Census, the 2011 Census and National Household Survey, as well as the latest available data from the 2016 Census. Additional data for the 2016 Census will be included in Censusmapper within a day or two after public release by Statistics Canada. Statistics Canada maintains a release schedule for the Census 2016 Program which can be viewed on their website.

To view available datasets, run `list_census_datasets()`.

```
list_census_datasets()
#> # A tibble: 3 x 2
#>   dataset                description
#> *   <chr>                      <chr>
#> 1    CA06          2006 Canada Census
#> 2    CA11 2011 Canada Census and NHS
#> 3    CA16          2016 Canada Census
```

As other Census datasets become available via the CensusMapper API, they will be listed as output when calling `list_census_datasets()`.

Census Regions

Census data is aggregated at multiple geographic levels. Census geographies at the national (C), provincial (PR), census metropolitan area (CMA), census division (CD), and census subdivision (CSD) are defined as named census regions.

Canadian Census geography can change in between Census periods. **cancensus** provides a function, `list_census_regions(dataset)`, to display all named census regions and their corresponding id for a given census dataset.

```
list_census_regions("CA16")
#> # A tibble: 5,504 x 8
#>    region                      name level      pop municipal_status
#>     <chr>                     <chr> <chr>    <int>            <chr>
#>  1     01                    Canada     C 35151728             <NA>
#>  2     35                   Ontario    PR 13448494             <NA>
#>  3     24                    Quebec    PR  8164361             <NA>
#>  4     59          British Columbia    PR  4648055             <NA>
#>  5     48                   Alberta    PR  4067175             <NA>
#>  6     46                  Manitoba    PR  1278365             <NA>
#>  7     47             Saskatchewan    PR  1098352            <NA>
#>  8     12              Nova Scotia    PR   923598            <NA>
#>  9     13            New Brunswick    PR   747101            <NA>
#> 10     10 Newfoundland and Labrador    PR   519716            <NA>
#> # ... with 5,494 more rows, and 3 more variables: CMA_UID <int>,
#> #   CD_UID <int>, PR_UID <int>
```

The `regions` parameter in `get_census` requires as input a list of region id strings that correspond to that regions geoid. You can combine different regions together into region lists.

```
# Retrieves Vancouver and Toronto
list_census_regions('CA16') %>%
  filter(level == "CMA", name %in% c("Vancouver","Toronto"))
census_data <- get_census(dataset='CA16', regions=list(CMA=c("59933","35535")),
                          vectors=c("v_CA16_408","v_CA16_409","v_CA16_410"),
                          level='CSD', use_cache = FALSE)
```

Census Geographic Levels

Census data accessible through **cancensus** comes is available in a number of different aggregation levels including:

Code	Description	Count in Census 2016
C	Canada (total)	1
PR	Provinces/Territories	13
CMA	Census Metropolitan Area	49
CD	Census Division	287
CSD	Census Subdivision	713
CT	Census Tracts	5621
DA	Dissemination Area	56589
Regions	Named Census Region	

Selecting `regions = "59933"` and `level = "CT"` will return data for all 478 census tracts in the Vancouver Census Metropolitan Area. Selecting `level = "DA"` will return data for all 3450 dissemination areas. Working with CT and DA level data significantly increases the size of data downloads and API usage. **cancensus** relies on local data caching to reduce usage and load times.

Setting `level = "Regions"` will produce data strictly for the selected region without any tiling of data at lower census aggregation levels levels.

Working with Census Variables

Census data contains thousands of different geographic regions as well as thousands of unique variables. In addition to enabling programmatic and reproducible access to Census data, **cancensus** has a number of tools to help users find the data they are looking for.

Displaying available Census variables

Run `list_census_vectors(dataset)` to view all available Census variables for a given dataset.

```
list_census_vectors("CA16")
#> # A tibble: 6,611 x 7
#>        vector   type                                          label  units
#>         <chr> <fctr>                                          <chr> <fctr>
#>  1 v_CA16_401  Total                               Population, 2016 Number
#>  2 v_CA16_402  Total                               Population, 2011 Number
#>  3 v_CA16_403  Total    Population percentage change, 2011 to 2016 Number
#>  4 v_CA16_404  Total                        Total private dwellings Number
#>  5 v_CA16_405  Total Private dwellings occupied by usual residents Number
#>  6 v_CA16_406  Total       Population density per square kilometre  Ratio
#>  7 v_CA16_407  Total                 Land area in square kilometres Number
#>  8   v_CA16_1  Total                                    Total - Age Number
#>  9   v_CA16_2   Male                                    Total - Age Number
#> 10   v_CA16_3 Female                                    Total - Age Number
#> # ... with 6,601 more rows, and 3 more variables: parent_vector <chr>,
#> #   aggregation <chr>, details <chr>
```

Variable characteristics

For each variable (vector) in that Census dataset, this shows:

- Vector: short variable code
- Type: variables are provided as aggregates of female responses, male responses, or total (male+female) responses
- Label: detailed variable name
- Units: provides information about whether the variable represents a count integer, a ratio, a percentage, or a currency figure
- Parent_vecctor: shows the immediate hierarchical parent category for that variable, where appropriate
- Aggregation: indicates how the variable should be aggregated with others, whether it is additive or if it is an average of another variable
- Description: a rough description of a variable based on its hierarchical structure. This is constructed by **cancensus** by recursively traversing the labels for every variable's hierarchy, and facilitates searching for specific variables using key terms.

Variable search

Each Census dataset features numerous variables making it a bit of a challenge to find the exact variable you are looking for. To help with that, this package includes a built-in vector search tool to help find specific variables.

```
# Find the variable indicating the number of people of Austrian ethnic origin
search_census_vectors("Austrian", dataset = 'CA16')
#> # A tibble: 3 x 7
#>        vector   type    label  units parent_vector aggregation
#>         <chr> <fctr>    <chr> <fctr>         <chr>       <chr>
#> 1 v_CA16_4092  Total Austrian Number   v_CA16_4089    Additive
#> 2 v_CA16_4093   Male Austrian Number   v_CA16_4090    Additive
#> 3 v_CA16_4094 Female Austrian Number   v_CA16_4091    Additive
#> # ... with 1 more variables: details <chr>
```

Knowing exactly how to spell the right variable can be tricky, but this search function relies on fuzzy searching so if it is unable to find an exact match of your search term, it will provide some helpful alternatives. In this case, searching for “Austraian” origin will show search terms for the vectors for both Austrian and Australian origins.

```
# Find the variable indicating the number of people of Austrian ethnic origin
search_census_vectors("Austraian", dataset = 'CA16')
#> # A tibble: 2 x 1
#>   `Similarly named objects`
#>                       <chr>
#> 1                  Austrian
#> 2                Australian
```

Managing variable hiararchy

Census variables are frequently hierarchical. As an example, consider the variable for the number of people of Austrian ethnic background. We can select that vector and quickly look up its entire hierarchy using `parent_census_vectors` on a vector list.

```
list_census_vectors("CA16") %>%
  filter(vector == "v_CA16_4092") %>%
  parent_census_vectors()
#> # A tibble: 3 x 7
#>        vector   type
#>         <chr> <fctr>
#> 1 v_CA16_4089  Total
#> 2 v_CA16_4044  Total
#> 3 v_CA16_3999  Total
#> # ... with 5 more variables: label <chr>, units <fctr>,
#> #   parent_vector <chr>, aggregation <chr>, details <chr>
```

Sometimes we want to traverse the hierarchy in the oppposite direction. This is frequently required when looking to compare different variable stems that share the same aggregate variable. As an example, if we want to look the total count of Northern European ethnic origin respondents disaggregated by individual countries, it is pretty easy to do so.

```
# Find the variable indicating the Northern European aggregate
search_census_vectors("Northern European", dataset = 'CA16')
#> # A tibble: 6 x 7
#>        vector   type
#>         <chr> <fctr>
#> 1 v_CA16_4122  Total
#> 2 v_CA16_4123   Male
#> 3 v_CA16_4124 Female
#> 4 v_CA16_4140  Total
#> 5 v_CA16_4141   Male
#> 6 v_CA16_4142 Female
#> # ... with 5 more variables: label <chr>, units <fctr>,
#> #   parent_vector <chr>, aggregation <chr>, details <chr>
```

The search result shows that the vector **v_CA16_4092** represents the aggregate for all Northern European origins. The `child_census_vectors` function can return a list of its constituent underlying variables.

```
# Show all child variable leaves
list_census_vectors("CA16") %>%
  filter(vector == "v_CA16_4122") %>% child_census_vectors(leaves = TRUE)
#> # A tibble: 6 x 7
#>        vector   type                             label  units
#>         <chr> <fctr>                               <chr> <fctr>
#> 1 v_CA16_4125  Total                              Danish Number
#> 2 v_CA16_4128  Total                             Finnish Number
#> 3 v_CA16_4131  Total                           Icelandic Number
#> 4 v_CA16_4134  Total                           Norwegian Number
#> 5 v_CA16_4137  Total                             Swedish Number
#> 6 v_CA16_4140  Total Northern European origins, n.i.e. Number
#> # ... with 3 more variables: parent_vector <chr>, aggregation <chr>,
#> #   details <chr>
```

The `leaves = TRUE` parameter specifies whether intermediate aggregates are included or not. If `TRUE` then only the lowest level variables are returns - the "leaves" of the hierarchical tree.