Package 'easycensus'

October 13, 2022

Title Quickly Find, Extract, and Marginalize U.S. Census Tables

Version 1.0.0

Description Extracting desired data using the proper Census variable names can be time-consuming. This package takes the pain out of that process by providing functions to quickly locate variables and download labeled tables from the Census APIs (<https://www.census.gov/data/developers/data-sets.html>).

Depends R (>= 2.10)

Imports rlang, vctrs, pillar, dplyr (>= 1.0.0), tidyr (>= 1.0.0), stringr, censusapi, cli

Suggests posterior, testthat (>= 3.0.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

URL https://corymccartan.com/easycensus/, https://github.com/CoryMcCartan/easycensus/

BugReports https://github.com/CoryMcCartan/easycensus/issues

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Cory McCartan [aut, cre]

Maintainer Cory McCartan <cmccartan@g.harvard.edu>

Repository CRAN

Date/Publication 2022-08-25 17:42:35 UTC

R topics documented:

`cens_auth` *Authorize use of the Census API*

Description

Tries environment variables `CENSUS_API_KEY` and `CENSUS_KEY`, in that order. If none is found and R is used in interactive mode, will prompt the user for a key.

Usage

```
cens_auth()
```

Value

a Census API key

`cens_find` *Find a decennial or ACS census table with variables of interest*

Description

This function uses fuzzy matching to help identify tables from the census which contain variables of interest. Matched table codes are printed out, along with the Census-provided table description, the parsed variable names, and example table cells. The website `https://censusreporter.org/` may also be useful in finding variables.

Usage

```
cens_find_dec(..., show = 2)

cens_find_acs(..., show = 4)
```

Arguments

...	Variables to look for. These can be length-1 character vectors, or, for convenience, can be left unquoted (see examples).
show	How many matching tables to show. Increase this to show more possible matches, at the cost of more output. Negative values will be converted to positive but will suppress any printing.

Value

The codes for the top show tables, invisibly if show is positive.

Examples

```
cens_find_dec("sex", "age")
cens_find_dec(tenure, race)
cens_find_acs("income", "sex", show=3)
cens_find_acs("heath care", show=-1)
```

---

cens_geo *Construct a Geography Specification for Census Data*

Description

Currently used mostly internally. Builds a Census API-formatted specification of which geographies to download data for. State and county names (or postal abbreviations) are partially matched to existing tables, for ease of use. Other geographies should be specified with Census GEOIDs. The usgazeteer package, available with `remotes::install_github("bhaskarvk/usgazetteer")`, may be useful in finding GEOIDs for other geographies. Consult the "geography" sections of each API at https://www.census.gov/data/developers/data-sets.html for information on which geographic specifiers may be provided in combination with others.

Usage

```
cens_geo(geo = NULL, ..., check = TRUE, api = "acs/acs5", year = 2019)
```

Arguments

geo	The geographic level to return. One of the machine-readable or human-readable names listed in the "Details" section. Will return all matching geographies of this level, as filtered by the further arguments to .... For example, setting geo="tract" is equivalent to setting tract="all".
...	Geographies to return, as supported by the Census API. Order matters here—the first argument will be the geographic level to return (i.e., it corresponds to the geo argument) and additional arguments will filter the results. Use "all", "*", NA, or TRUE to return all units of a particular geography. See the examples for details.

check	If TRUE, validate the provided geographies against the available geographies from the relevant Census API. Requires the api and year arguments to be specified.
api	A Census API programmatic name such as "acs/acs5".
year	The year for the data

Details

Supported geography arguments:

- us
- region
- division
- state
- county
- county_subdiv (County Subdivision)
- subminor_civil_division (Subminor Civil Division)
- place_remainder (Place/Remainder (Or Part))
- tract_part (Tract (Or Part))
- urban_rural (Urban Rural)
- block_group_part (Block Group (Or Part))
- block
- tract
- aian_area_part (American Indian Area/Alaska Native Area/Hawaiian Home Land (Or Part))
- block_group (Block Group)
- county_part (County (Or Part))
- place_part (Place (Or Part))
- place
- consolidated_city (Consolidated City)
- alaska_native_regional_corporation (Alaska Native Regional Corporation)
- aian_area (American Indian Area/Alaska Native Area/Hawaiian Home Land)
- tribal_subdiv (Tribal Subdivision/Remainder)
- aian_reserve_stat (American Indian Area/Alaska Native Area (Reservation Or Statistical Entity Only))
- ai_tribal_subdiv_part (American Indian Tribal Subdivision (Or Part))
- ai_off_reserve_trust (American Indian Area (Off-Reservation Trust Land Only)/Hawaiian Home Land)
- tribal_census_tract (Tribal Census Tract)
- tribal_census_tract_part (Tribal Census Tract (Or Part))
- tribal_block_group (Tribal Block Group)

- state_part (State (Or Part))
- county_subdiv_part (County Subdivision (Or Part))
- tribal_subdiv_part (Tribal Subdivision/Remainder (Or Part))
- aian_reserve_stat_part (American Indian Area/Alaska Native Area (Reservation Or Statistical Entity Only) (Or Part))
- ai_off_reserve_trust_part (American Indian Area (Off-Reservation Trust Land Only)/Hawaiian Home Land (Or Part))
- tribal_block_group_part (Tribal Block Group (Or Part))
- msa (Metropolitan Statistical Area/Micropolitan Statistical Area)
- principal_city_part (Principal City (Or Part))
- metro_division (Metropolitan Division)
- msa_part (Metropolitan Statistical Area/Micropolitan Statistical Area (Or Part))
- metro_division_part (Metropolitan Division (Or Part))
- combined_statistical_area (Combined Statistical Area)
- combined_necta (Combined New England City And Town Area)
- necta (New England City And Town Area)
- combined_statistical_area_part (Combined Statistical Area (Or Part))
- combined_necta_part (Combined New England City And Town Area (Or Part))
- necta_part (New England City And Town Area (Or Part))
- principal_city (Principal City)
- necta_division (Necta Division)
- necta_division_part (Necta Division (Or Part))
- urban_area (Urban Area)
- urban_area_part (Urban Area (Or Part))
- consolidated_city_part (Consolidated City (Or Part))
- cd (Congressional District)
- sld_upper (State Legislative District (Upper Chamber))
- sld_lower (State Legislative District (Lower Chamber))
- alaska_native_regional_corporation_part (Alaska Native Regional Corporation (Or Part))
- zcta (Zip Code Tabulation Area)
- zcta_part (Zip Code Tabulation Area (Or Part))
- school_district_elementary (School District (Elementary))
- school_district_secondary (School District (Secondary))
- school_district_unified (School District (Unified))
- ai_tribal_subdiv (American Indian Tribal Subdivision)
- puma (Public Use Microdata Area)

Value

A list with two elements, `region` and `regionin`, which together specify a valid Census API geography argument.

Examples

```
cens_geo(state="WA")
cens_geo("county", state="WA") # equivalent to `cens_geo(county="all", state="WA")`
cens_geo(county="King", state="Wash")
cens_geo(zcta="02138", check=FALSE)
cens_geo(zcta=NA, state="WA", check=FALSE)
cens_geo("zcta", state="WA", check=FALSE)
cens_geo(cd="09", state="WA", check=FALSE)
cens_geo("county_part", state="WA", cd="09", check=FALSE)
```

---

cens_get *Download data from a decennial census or ACS table*

Description

Leverages `censusapi::getCensus()` to download tables of census data. Tables are returned in tidy format, with variables given tidy, human-readable names.

Usage

```
cens_get_dec(
  table,
  geo = NULL,
  ...,
  check_geo = FALSE,
  drop_total = FALSE,
  show_call = FALSE
)

cens_get_acs(
  table,
  geo = NULL,
  ...,
  year = 2019,
  survey = c("acs5", "acs1"),
  check_geo = FALSE,
  drop_total = FALSE,
  show_call = FALSE
)
```

Arguments

table	The table to download, either as a character vector or a table object as produced by cens_find_dec(), cens_find_acs() or cens_parse_tables(), or as included in tables_dec and tables_acs. Note: some tables are split into A/B/C/etc. versions by race; this function unifies all of these tables under one code. So, for example, use P012, not P012A.
geo	The geographic level to return. One of the machine-readable or human-readable names listed in the "Details" section of cens_geo(). Will return all matching geographies of this level, as filtered by the further arguments to .... For example, setting geo="tract" is equivalent to setting tract="all".
...	Geographies to return, as supported by the Census API. Order matters here—the first argument will be the geographic level to return (i.e., it corresponds to the geo argument) and additional arguments will filter the results. Use "all", "*", NA, or TRUE to return all units of a particular geography. See the examples of cens_geo() for details.
check_geo	If TRUE, validate the provided geographies against the available geographies from the relevant Census API.
drop_total	Whether to filter out variables which are totals across another variable. Recommended only after inspection of the underlying table.
show_call	Whether to show the actual call to the Census API. May be useful for debugging.
year	For ACS data, the survey year to get data for.
survey	For ACS data, whether to use the one-year or five-year survey (the default). Make sure to check availability using cens_find_acs().

Value

A tibble of census data in tidy format, with columns GEOID, NAME, variable (containing the Census variable code), value or estiamte, moe in the case of ACS tables, and additional factor columns specific to the table.

Examples

```
## Not run:
cens_get_dec("P3", "state")
cens_get_dec(tables_sf1$H2, "state")
cens_get_dec("H2", "county", state="WA", drop_total=TRUE)

cens_get_acs("B09001", county="King", state="WA")

## End(Not run)
```

cens_margin_to *Helper function to sum over nuisance variables*

Description

For ACS data, margins of error will be updated appropriately, using the functionality in `estimate()`.

Usage

```
cens_margin_to(data, ...)
```

Arguments

data	The output of `cens_get_dec()` or `cens_get_acs()`
...	The variables of interest, which will be kept. Remaining variables will be marginalized out.

Value

A new data frame that has had `group_by()` and `summarize()` applied.

Examples

```
## Not run:
d_cens = cens_get_acs("state", "B25042")
cens_margin_to(d_cens, bedrooms)

## End(Not run)
```

cens_parse_tables *Attempt to Parse Tables from a Census API*

Description

Uses the same parsing code as that which generates tables_sf1 and tables_acs See `https://www.census.gov/data/developers/data-sets.html` for a list of APIs and corresponding years, or use `censusapi::listCensusApis()`.

Usage

```
cens_parse_tables(api, year)
```

Arguments

api	A Census API programmatic name such as `"acs/acs5"`.
year	The year for the data

Value

A list of `cens_table` objects, which are just lists with four elements:

- `concept`, a human-readable name
- `tables`, the constituent table codes
- `surveys`, the supported surveys
- `dims`, the parsed names of the dimensions of the tables
- `vars`, a `tibble` with all of the parsed variable values

Examples

```
## Not run:
cens_parse_tables("dec/pl", 2020)

## End(Not run)
```

`estimate` *Estimate class*

Description

A numeric vector that stores margin-of-error information along with it. The margin of error will update through basic arithmetic operations, using a first-order Taylor series approximation. The implicit assumption is that the errors in each value are uncorrelated. If in fact there is correlation, the margins of error could be wildly under- or over-estimated.

Usage

```
estimate(x, se = NULL, moe = NULL, conf = 0.9)

is_estimate(x)

as_estimate(x)
```

Arguments

`x`	A numeric vector containing the estimate(s).
`se`	A numeric vector containing the standard error(s) for the estimate(s). Users should supply either `se` or `moe` and `conf`.
`moe`	A numeric vector containing the margin(s) of error. Users should supply either `se` or `moe` and `conf`.
`conf`	The confidence level to use in converting the margin of error to a standard error. Defaults to 90%, which is what the Census Bureau uses for ACS estimates.

Value

An `estimate` vector.

Examples

```
estimate(5, 2) # 5 with std. error  2
estimate(15, moe=3) - estimate(5, moe=4)
estimate(1:4, 0.1) * estimate(1, 0.1)
```

`est_prop` *Specialized margin-of-error calculations*

Description

Proportions and percent-change-over-time calculations require different standard error calculations.

Usage

```
est_prop(x, y)

est_pct_chg(x, y)
```

Arguments

`x, y`	An estimate vector. For `est_pct_chg()`, calculates the % change from `x` to `y` (i.e., $(y - x)/x$)

Value

An estimate vector.

Examples

```
x = estimate(1, 0.1)
y = estimate(1.5, 0.1)
est_prop(x, y)
est_pct_chg(x, y)
```

format.estimate *Format an estimate*

Description

Format an estimate for pretty printing

Usage

```
## S3 method for class 'estimate'
format(x, conf = 0.9, digits = 2, trim = FALSE, ..., formatter = fmt_plain)
```

Arguments

x	An estimate vector
conf	The confidence level to use in converting the margin of error to a standard error. Defaults to 90%, which is what the Census Bureau uses for ACS estimates.
digits	The number of dig
trim	logical; if FALSE, logical, numeric and complex values are right-justified to a common width: if TRUE the leading blanks for justification are suppressed.
...	Ignored.
formatter	the formatting function to use internally

get_est *Extract estimates, standard errors, and margins of error*

Description

Getter functions for `estimate()` vectors.

The posterior::rvar class may be useful in handling standard errors for more complicated mathematical expressions. This function assumes a Normal distribution centered on the estimate, with standard deviation equal to the standard error of the estimate. The `posterior` package is required for this function.

Usage

```
get_est(x)

get_se(x)

get_moe(x, conf = 0.9)

to_rvar(x, n = 500)
```

Arguments

x	An estimate vector.
conf	The confidence level to use in constructing the margin of error.
n	How many samples to draw.

Value

An estimate vector.

A posterior::rvar vector.

Examples

```
x = estimate(1, 0.1)
get_est(x)
get_moe(x)

x = estimate(1, 0.1)
if (requireNamespace("posterior", quietly=TRUE)) {
    rv_x = to_rvar(x)
    (rv_x^2 / rv_x) - rv_x # std. errors zero (correct)
    x^2 / x - x # std. errors not zero
}
```

---

tables *Parsed Census SF1 and ACS Tables*

Description

Contains parsed table information for the 2010 Decennial Summary File 1 and 2019 ACS 5-year and 1-year tables. This parsed information is used internally in `cens_find_dec()`, `cens_find_acs()`, `cens_get_dec()`, and `cens_get_acs()`. For other sets of tables, try using `cens_parse_tables()`.

Usage

```
tables_sf1

tables_acs
```

Format

A list of `cens_table` objects, which are just lists with four elements:

- `concept`, a human-readable name
- `tables`, the constituent table codes
- `surveys`, the supported surveys
- `dims`, the parsed names of the dimensions of the tables

- vars, a tibble with all of the parsed variable values

An object of class list of length 83.

An object of class list of length 848.

---

tidiers *Tidy labels in census tables*

Description

Some table labels are quite verbose, and users will often want to shorten them. These functions make tidying common types of labels easy. Most produce straightforward output, but there are several more generic tidiers:

- tidy_simplify() attempts to simplify labels by removing words common to all labels.
- tidy_parens() attempts to simplify labels by removing all terms in parentheses.
- tidy_race_detailed() creates logical columns for each of the six racial categories.

Usage

```
tidy_race(x)

tidy_race_detailed(x, x2, x3)

tidy_ethnicity(x)

tidy_age(x)

tidy_age_bins(x, as_factor = FALSE)

tidy_income_bins(x, as_factor = FALSE)

tidy_simplify(x)

tidy_parens(x)
```

Arguments

x	A factor, which will be re-leveled. Character vectors will be converted to factors.
x2, x3	Additional character columns containing detailed information for certain variables (e.g. detailed race)
as_factor	if TRUE, return a factor with levels of the form [35,40].

Value

A re-leveled factor, except for tidy_age_bins(), which by default returns a data frame with columns age_from and age_to (inclusive).

Examples

```
ex_race_long = c("american indian and alaska native alone", "asian alone",
    "black or african american alone", "hispanic or latino",
    "native hawaiian and other pacific islander alone",
    "some other race alone", "total", "two or more races",
    "white alone", "white alone, not hispanic or latino")
tidy_race(ex_race_long)

tidy_age_bins(c("10 to 14 years", "21 years", "85 years and over"))

tidy_parens(c("label one (fake)", "label two (fake)"))
tidy_simplify(c("label one (fake)", "label two (fake)"))

## Not run:  # requires API key
d = cens_get_acs("B02003", "us", year=2019, survey="acs1")
dplyr::mutate(d, tidy_race_detailed(dtldr_1, dtldr_2, dtldr_3))

## End(Not run)
```

Index