Package 'censusprofiler'

March 25, 2024

Type Package

Title censusprofiler

Version 0.21

Date 2024-3-25

Author RW Stevenson

Maintainer RW Stevenson <thebobstevenson@protonmail.com>

Imports httr,tidyr,dplyr, sf, stringr, tigris, units, tmap, tmaptools,officer,flextable,formattable,jsonlite,paletteer,tibble,purrr,ggplot2,data.table

Depends R (>= 3.5.0)

Description Package censusprofiler is designed to simplify regionalized census data capture. While censusprofiler can perform several functions, at its core, it takes a geographic point, draws a radius, and makes calls to the census api for geographical units around within that radius, and provides output that is more suited for presentation. All census calls are made via the US Census API. This package was designed to interface with the American Community Survey in mind. However, the US Census API provides multiple datasets. Implementation of these additional data sources remains a TODO for the package.

License GPL (>= 2)

RoxygenNote 7.3.1

Encoding UTF-8

Collate 'capi.R'
'mapper.R'
'profiler.R'
'tabler.R'
'utility_data.R'
'utility_geo.R'

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/rws-r/censusprofiler

BugReports https://github.com/rws-r/censusprofiler/issues

R topics documented:

`capi` *Census API Data Call*

Description

An API interface for capturing and formatting census data.

Usage

```
capi(
  year = NULL,
  tableID = NULL,
  variables = NULL,
  geography = NULL,
  filterAddress = NULL,
  filterRadius = NULL,
  ggr = NULL,
  geosObject = NULL,
  mode = "table",
  filterSummary = FALSE,
  filterSummaryLevels = "root",
  filterByGeoType = NULL,
  filterByGeoValue = NULL,
  state = NULL,
  county = NULL,
```

```
  tract = NULL,
  block_group = NULL,
  place = NULL,
  metro = NULL,
  consolidatedCity = NULL,
  region = NULL,
  division = NULL,
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
  censusVars = NULL,
  verbose = FALSE,
  profile = FALSE,
  fast = FALSE,
  simpleReturn = FALSE,
  test = FALSE,
  st = NULL
)
```

Arguments

year	Year for data call.
tableID	Formerly known as varBase, or concept, or group: i.e., "B01001"
variables	A vector of variables for the call. If multiple select variables per tableID are desired, then variables should be constructed as a named list, with tableID as name, and sub list items as variables—either full ("B01001_001") or numeric (c(1:8)).
geography	Specifying geography: e.g., "tract", "county"
filterAddress	An address input used to generate a radius around, for filtering data.
filterRadius	A numeric value specifying the radius in miles around the address.
ggr	Internal: to pass a get_geocode_radius() object to function.
geosObject	Optional, attach geos object to simplify geo processes.
mode	c("table","summarize","median")
filterSummary	Logical parameter to specify whether to filter out summary levels (typically _001 and therefore "root").
filterSummaryLevels	Explicit description of lowest type denoting summary level. Also excludes lower levels.
filterByGeoType	An irregular geo type to get a smaller overlapping set of tracts, block_groups or other geography from. Options are currently "metro", "place","combined_statistical_areas". E.g., Find all tracts in Chicago (place).
filterByGeoValue	A value to find object for filtering. Either NAME or GEOID.
state	Input (abb. or FIPS) of state for search.
county	Input (abb. or FIPS) of county for search.
tract	Input (abb. or FIPS) of tract for search.
block_group	Input (abb. or FIPS) of block group for search.

place	Input (abb. or FIPS) of place for search.
metro	Input (abb. or FIPS) of metropolitan statistical area for search.
consolidatedCity	Input (abb. or FIPS) of consolidated city for search.
region	Input (abb. or FIPS) of region for search.
division	Input (abb. or FIPS) of division for search.
dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables
verbose	Logical parameter to specify whether to produce verbose output.
profile	Logical parameter to specify whether to build profile.
fast	Internal parameter for pseudo_tableID and stat table
simpleReturn	Param to return raw data, not formatted.
test	Internal parameter for testing suite.
st	Internal parameter to provide timestamp consistency.

Value

dataframe

Examples

```
## Not run:
Basic call
capi(year=2022,datatype="acs", dataset="acs5", tableID="B01001",
variables=c("B01001_001","B01001_002"), geography="tract", filterAddress=v,
filterRadius=1, ggr=NULL, mode="table", filterSummary=FALSE,
filterSummaryLevels="root", state=NULL, county=NULL, tract=NULL,
block_group=NULL, verbose=TRUE, profile=FALSE, st=NULL)

capi(year=2022,datatype="acs", dataset="acs5", tableID="B01001",
variables=c("B01001_001","B01001_002"), geography="tract", filterAddress=v,
filterRadius=1, ggr=NULL, mode="summarize", filterSummary=FALSE,
filterSummaryLevels="root", state=NULL, county=NULL, tract=NULL,
block_group=NULL, verbose=TRUE, profile=FALSE, st=NULL)

## End(Not run)
```

---

comparison_helper *Comparison Helper*

Description

Comparison Helper

Usage

```
comparison_helper(
  df = NULL,
  comparisonDF = NULL,
  comp_type = NULL,
  tableID = NULL,
  stateFilter = NULL,
  verbose = FALSE
)
```

Arguments

df	A primary dataset to provide comparisons against.
comparisonDF	A geography-level comparison dataset.
comp_type	Explicitly call comparison type.
tableID	The tableID to reference comparison.
stateFilter	FIPS or abbreviation for filtering and selecting comparisons.
verbose	Logical parameter to specify whether to produce verbose output.

Value

Dataframe

Examples

```
## Not run:
comparison_helper(data,comparisondata,tableID="B02001",stateFilter=17)

## End(Not run)
```

---

create_comparison_data

Create Profile (Batch By Geographies)

Description

This function relies on create_profile_batch for its internal logic, but is used primarily to create comparison profile objects for whole geographies. This is useful when rendering displayTable() and a larger geography comparison is desired.

Usage

```
create_comparison_data(
  geography = NULL,
  profileDataset = NULL,
  year = NULL,
  variables = NULL,
  tableID = NULL,
  coordColName = "sf",
  verbose = FALSE,
```

```
  geosObject = NULL,
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
  censusVars = NULL,
  test = FALSE
)
```

Arguments

geography	Either "us" or "state".
profileDataset	Optional dataset to determine state/county for filtering.
year	Numeric value specifying year of ACS call.
variables	A variables vector.
tableID	A tableID vector.
coordColName	Default, set to "sf", but can be changed if non-sf object.
verbose	Whether to provide verbose output.
geosObject	Optional geosObject to speed up processing time.
dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables
test	Internal: for testing purposes.

Value

A deep, nested list. Structure is list1 > regionInfo,data. regionInfo is a tibble with identifiers, and data contains three tibbles: df, dfCount, dfNoSummary. See create_profile for more information. To access individual datasets, use this structure: object$data[[n]]$profileTable.

Examples

```
## Not run:
create_comparison_data(geography="state",
year=2021,variables = profile_variables,tableID = profile_tableID)
create_comparison_data(geography="us",year=2021,
variables = profile_variables,tableID = profile_tableID)

## End(Not run)
```

---

entropyIndex *Entropy Index*

Description

A statistical function to estimate diversity / segregation in datasets.

Usage

```
entropyIndex(
  data = NULL,
  dataType = "long",
  dissimilarityValue = NULL,
  dissimilarityValueB = NULL,
  geography = "tract",
  wideCols = NULL,
  longCol = "pct",
  filterSummary = FALSE,
  filterSummaryLevels = "root",
  tableID = NULL,
  variables = NULL,
  filterAddress = NULL,
  filterRadius = NULL,
  state = NULL,
  county = NULL,
  tract = NULL,
  block_group = NULL,
  year = NULL,
  return = FALSE,
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
  censusVars = NULL,
  verbose = FALSE
)
```

Arguments

data	A data object for which to estimate entropy.
dataType	Can choose "long", "wide" or "vector" depending on data object type.
dissimilarityValue	For dissimilarity index, selecting minority group. Numeric value corresponding to variable.
dissimilarityValueB	For exposure/isolation index, selecting minority group. Numeric value corresponding to variable.
geography	Defaults to "tract," but must match data object.
wideCols	Specified columns for calculating entropy in wide data.
longCol	Specified columns for calculating entropy in long data.
filterSummary	Logical parameter to specify whether to filter out summary levels (typically _001 and therefore "root").
filterSummaryLevels	Explicit description of lowest type denoting summary level. Also excludes lower levels.#'
tableID	TableID specification for calculating entropy.
variables	Variable specification for calculating entropy.
filterAddress	For data calls, a filtered area specification.

filterRadius	For data calls, a filtered area specification.
state	Input (abb. or FIPS) of state for search.
county	Input (abb. or FIPS) of county for search.
tract	Input (abb. or FIPS) of tract for search.
block_group	Input (abb. or FIPS) of block group for search.
year	Year for data call.
return	Logical parameter. If TRUE, return value only. Otherwise return formatted string.
dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables #' @param verbose Logical parameter to specify whether to produce verbose output.

Details

Logic developed from https://www2.census.gov/programs-surveys/demo/about/housing-patterns/multigroup_entropy.pd and Scientific Study of Religion - 2016 - Dougherty - Congregational Diversity and Attendance in a Mainline Protestant-2024-02-19-13-18.pdf OR https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1468-5906.2008.00390.x."EI" is built from the second, while MGEI is built from the former. Basically, the EI signifies how much diversity exists in a community (e.g., Census Tract). The Multigroup index of the whole metro area examines how much segregation exist between areas in a larger area. It doesn't account for the diversity of the whole, but the integration of the areas. So for example, Individual census tracts may have a high entropy score (0.8), signifying lots of diversity. But it may have a low metro score (.05) suggesting that diversity is evenly spread throughout the larger area. If the metro MGEI was higher, it would signify more segregation between diverse areas.

Value

dataframe

Examples

```
## Not run:
entropyIndex(data=NULL,tableID = "B11012",variables =
c(1:4),year=2022,verbose=TRUE,filterAddress = v,filterRadius = 1)

## End(Not run)
```

---

geocoder *Geocoder*

Description

Using Census Geocoder

Usage

```
geocoder(address, year = 2020, service = "census", verbose = FALSE)
```

Arguments

address	Address entry in STREET, CITY, STATE ZIP format.
year	Year to get census geocoding. See census info for year specs.
service	Whether to use census geocoder or OSM.
verbose	Logical param to provide feedback.

Value

Dataframe with coordinates.

Examples

```
## Not run:
#Using census data
geocoder(address="350 Fifth Avenue New York, NY 10118",year=2020,service='census')

#Using OSM data
geocoder(address="350 Fifth Avenue New York, NY 10118",service='OSM')

## End(Not run)
```

---

geocoder_batch *Geocoder Batch*

Description

Geocoder Batch

Usage

```
geocoder_batch(
  addressList,
  addressCol = "address",
  start = NULL,
  limit = NULL,
  verbose = FALSE
)
```

Arguments

addressList	List of addresses to geocode.
addressCol	Name or number of address column.
start	If needed, a starting ID to begin batch processing.
limit	If needed, an ending ID to end batch processing.
verbose	Logical param to provide feedback.

Value

A modified dataframe including coordinates.

Examples

```
## Not run:
geocoder_batch(addressList)
geocoder_batch(addressList,start=10,limit=50)

## End(Not run)
```

`geo_road_helper` *Geo: Road Helper*

Description

A geo utility to assist filtering roads which do not meet certain criteria, to avoid bogging down tmap "plot" maps with too much text.

Usage

```
geo_road_helper(df = NULL, verbose = FALSE)
```

Arguments

`df`	The road sf object.
`verbose`	A logical parameter to specify verbose output.

Details

First, strips out all directional pre/suffixes. Then, returns a modified dataframe with added suffix column. Can use this to filter later.

Value

A dataframe with "suffix" column added, for filtering use later.

Examples

```
## Not run:
  roads <- tigris::roads(state=geo$states,county = geo$counties,year=2021,filter_by=bbox)
   roads <- geo_road_helper(roads)

## End(Not run)
```

`geo_var_builder` *Geography Variable Builder*

Description

Checks if geospatial objects exist for referencing census data. If not, get them.

Usage

```
geo_var_builder(
  geography = c("all"),
  try = "local",
  state = NULL,
  county = NULL,
  geosObject = NULL,
  verbose = FALSE,
  test = FALSE
)
```

Arguments

`geography`	Specifies the kind of geography to call. Options: "state","county","tract","block". Defaults to "all".
`try`	Parameter to specify whether to try local file read first, before downloading.
`state`	Filter by county name or COUNTYFP
`county`	Filter by state name or STATEFP
`geosObject`	Optional geosObject object to speed up geo processing.
`verbose`	Logical param to provide feedback.
`test`	Logical param to bypass feedback and messages during testing.

Value

Returns global objects if does not exist.

Examples

```
## Not run:
geo_var_builder()
geo_var_builder(geography="state")
geo_var_builder(geography="county")
geo_var_builder(geography="tract")
geo_var_builder(geography="block")

## End(Not run)
```

`get_census_variables` *Get Census Variables*

Description

Sends a JSON request to the Census API to capture variables for use in other functionality.

Usage

```
get_census_variables(
  year = NULL,
  dataset_main = NULL,
  dataset_sub = NULL,
  dataset_last = NULL,
  detailed_tagging = FALSE,
  directory = FALSE,
  verbose = FALSE
)
```

Arguments

`year`	Year for variable draw
`dataset_main`	Main dataset parameter (e.g., 'acs' or 'dec')
`dataset_sub`	Secondary dataset parameter (e.g., 'acs5')
`dataset_last`	Tertiary dataset parameter (e.g., 'cprofile' or 'subject')
`detailed_tagging`	Logical, to flag additional data for dataframe.
`directory`	Logical, to grab list of all datasets available in census API
`verbose`	Logical parameter to signal verbose output.

Value

A dataframe

Examples

```
## Not run:
get_census_variables(year=2022, dataset_main="acs", dataset_sub="acs5")

## End(Not run)
```

`get_geocode_radius` *Get a Geocoded Radius*

Description

A function to geocode a supplied address, and then return a specified list with various coordinate information.

Usage

```
get_geocode_radius(
  filterAddress = NULL,
  filterRadius = NULL,
  filterByGeoType = NULL,
  filterByGeoValue = NULL,
  state = NULL,
  county = NULL,
  geoidLookup = NULL,
  geography = NULL,
  radiusOnly = FALSE,
  geocodeOnly = FALSE,
  fipsOnly = FALSE,
  profile = FALSE,
  coords = NULL,
  intersectOverlap = 0.1,
  year = NULL,
  test = FALSE,
  verbose = FALSE,
  geosObject = NULL
)
```

Arguments

`filterAddress`	Address for geocoding. Should be in "Street, City, State Zip" format if possible. Names are also optional, but consistency of lookup is not guaranteed.
`filterRadius`	An integer describing the width of the radius in miles.
`filterByGeoType`	An irregular geo type to get a smaller overlapping set of tracts, block_groups or other geography from. Options are currently "metro", "place","combined_statistical_areas". E.g., Find all tracts in Chicago (place).
`filterByGeoValue`	A value to find object for filtering. Either NAME or GEOID.
`state`	Input (abb. or FIPS) of state for search.
`county`	Input (abb. or FIPS) of county for search.
`geoidLookup`	Lookup geography by GEOID.
`geography`	Typically passed by other functions, used for capi() request to specify geography. May be either "us","state", "county","tract", or "block group".
`radiusOnly`	Set to TRUE to return geocoded radius only, and not provide any additional ACS data requests.

geocodeOnly	Set to TRUE to return the geocode of the address only as an object.
fipsOnly	Logical param to return only FIPS from radius.
profile	A profile data object.
coords	Internal: passing already found coordinates to function.
intersectOverlap	A proportion of minimum required overlap from radius and unit area.
year	Year for data call
test	Internal: Parameter to allow testing passthroughs [depreciated]
verbose	Internal: Parameter for printing feedback.
geosObject	Optional geosObject object to speed up geo processing.

Details

Designed to work with censusprofiler functions, and using OpenStreetMap / US Census data, supplied addresses are geocoded. Resulting coordinates are processed, including radius buffers. Parameters are returned to the originating function, where capi() is called. Alternatively, it can be used to generate simply a geocoded address, or a geospatial radius.

Value

Returns either 1) a filtered sf object from capi() call; 2) an sf object with point coordinates; 3) an sf object with radius coordinates; or 4) a tmap map with radius overlay.

Examples

```
## Not run:
#Get full ACS dataset
get_geocode_radius(filterAddress="350 Fifth Avenue New York, NY 10118",
filterRadius=1,geometry=TRUE)

#Get the radius only.
get_geocode_radius(filterAddress="350 Fifth Avenue New York, NY 10118",
filterRadius=1,radiusOnly=TRUE)

#Get a simple geocode of an address.
get_geocode_radius(filterAddress="350 Fifth Avenue New York, NY 10118",
filterRadius=1,geocodeOnly=TRUE)

## End(Not run)
```

---

get_vre_table *Get VRE Tables*

Description

A function to calculate the margin of error based on aggregate ACS data.

Usage

```
get_vre_table(
  data = NULL,
  year = NULL,
  geography = NULL,
  tableID,
  variableList,
  variableAgg = FALSE,
  state = NULL,
  county = NULL,
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
  censusVars = NULL,
  verbose = FALSE,
  savePath = paste(getwd(), "/data/VRE/", sep = "")
)
```

Arguments

data	ACS data passed to function from capi().
year	ACS year passed to function from capi().
geography	ACS geography passed to function from capi().
tableID	ACS variable base passed to function from capi().
variableList	ACS variable number(s) passed to function from capi(). Integer or list.
variableAgg	Future TODO: make this suitable for a list of variables.
state	ACS state (descriptive or FIPS) passed to function from capi().
county	ACS county (descriptive or FIPS) passed to function from capi().
dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables
verbose	Logical parameter to display output
savePath	Local save path for VRE tables.

Details

The basic logic of this function is derived from the Census Bureau (https://www.census.gov/data/academy/webinars/2020 margins-of-error-acs.html). Information is stored in VRE tables on the CB FTP website in .csv files. This function finds, downloads, extracts, converts the .csv files, and then performs statistical calculations on the tables, for the aggregate data.

Value

Returns a dataframe with variance, standard error, and margin of error for aggregate data.

Examples

```
## Not run:
  VREMOE <- get_vre_table(data = df,year = 2021,geography = "tract",
  tableID = "B01001",variableList = c(2,3,3),variableAgg = NULL,state = "IL",
  county = 043)

## End(Not run)
```

---

load_data *Load Data*

Description

A convenience function that loads functional data, including ACS variables, a geos object, stats object, and/or geo profile comparison object. These are created and then if selected, loaded into the global environment.

Usage

```
load_data(
  load_censusVars = FALSE,
  load_geos = FALSE,
  load_stats = FALSE,
  load_profile_compare = FALSE,
  geography = "tract",
  geo_data = c("state", "county", "tract"),
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
  censusVars = NULL,
  loadToGlobal = FALSE,
  year = 2021,
  tableID = NULL,
  variables = NULL,
  geosObject = NULL,
  test = FALSE,
  verbose = FALSE
)
```

Arguments

load_censusVars	Logical param to capture census variables and concepts.
load_geos	Logical param to capture geos object.
load_stats	Logical param to create stats_object.
load_profile_compare	Logical param to create geo profile comparison object.
geography	Default geography option for stat_table_builder.
geo_data	Default geography options for load_geos.

dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables
loadToGlobal	Logical param to save to global environment.
year	Default year for data calls.
tableID	ProfileList object for stat_table_builder.
variables	Variable list for stat_table_builder.
geosObject	Optional geosObject to speed up processing time.
test	Internal logical parameter to specify testing envir.
verbose	Logical param to provide feedback.

Value

data.frame objects loaded into global environment.

Examples

```
## Not run:
load_data(load_acs=TRUE, load_geos=TRUE,load_stats=TRUE,
variables=profile_variables,tableID=profile_tableID)

## End(Not run)
```

---

mapper *mappeR*

Description

mappeR takes ACS data pulled from capi() and creates a map displaying estimates, either as simple counts, or percentages based on summary variables.

Usage

```
mapper(
  mapDF = NULL,
  tableID = NULL,
  variable = NULL,
  variableSummary = NULL,
  year = NULL,
  geography = NULL,
  filterAddress = NULL,
  filterRadius = NULL,
  coords = NULL,
  tract = NULL,
  county = NULL,
  state = NULL,
  geoidLookup = NULL,
  dispPerc = FALSE,
```

```
  LegendTitle = "Census Tracts",
  MapTitle = "Selected Census Tracts",
  radiusOnly = FALSE,
  areaOnly = FALSE,
  alpha = 0.3,
  interactive = FALSE,
  geosObject = NULL,
  markers = NULL,
  dispRoads = TRUE,
  dispWater = TRUE,
  dispPlaces = TRUE,
  dispRails = FALSE,
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
  censusVars = NULL,
  verbose = FALSE,
  test = FALSE,
  st = NULL
)
```

Arguments

mapDF	A census profiler object.
tableID	Variable base, prior to the underscore: i.e. "B01001" -» TODO Necessary???
variable	Since maps are not ideal for displaying a range of variables, the single variable number to display.
variableSummary	Summary variable against which to calculate the percentage.
year	Year for capi().
geography	Either "us","state","county","tract", or "block group".
filterAddress	Address for querying a set of geographies as a radius around a location.
filterRadius	The radius in miles.
coords	Parameter to pass coordinates to function for quicker geolocation.
tract	A list of specific tracts to filter.
county	County by name or FIPS code.
state	State by name or FIPS code.
geoidLookup	Lookup geography by GEOID.
dispPerc	Set to TRUE to calculate percentage based on summary variable.
LegendTitle	Set for custom title.
MapTitle	Text value to set map title.
radiusOnly	To display the tracts/counties/states and the radius on a map.
areaOnly	To display the tracts/counties/states within the area specified by geography.
alpha	Set alpha value of map.
interactive	Parameter for setting tmap_mode to either ‘interactive‘ or ‘plot.‘
geosObject	Optional geosObject to speed up processing time.

markers	A geos object containing sf files for roads, water, places or rails. Intended to speed up repeated uses of mapper, as in RMD files.
dispRoads	Logical: whether to display roads on non-interactive map.
dispWater	Logical: whether to display water bodies on non-interactive map.
dispPlaces	Logical: whether to display places on non-interactive map.
dispRails	Logical: whether to display railroads on non-interactive map.
dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables
verbose	Logical param to provide feedback.
test	Internal: Logical param to bypass feedback and provide limited data for testing.
st	Internal variable passed from other functions to provide consistency on timestamp.

Details

Logics include: 1) a call with variables + filterAddress (capi() call) [variable,variableSummary,filterAddress,filterRadius]
2) a call with variables + filterAddress + df included [mapDF,variable,variableSummary,filterAddress,filterRadius];
3) a call without variables + filterAddress displaying area by geography [filterAddress,filterRadius,areaOnly];
4) a call without variables + geography specified (e.g.: state=17, geography="county") [state/county/tract,areaOnly];
5) a call without variables + filterAddress and geography, but only radius displayed [filterAddress,filterRadius,radiusOnly]

Value

Returns a filtered tmap map, detailing either estimates or percentages of variables requested.

Examples

```
## Not run:
Create a map of a selected region based on a variable.
mapper(mapDF = NULL,tableID = "B02001",variable = "B02001_002",variableSummary =
"B02001_001",geography = "tract",year = 2022,dispPerc = FALSE,
MapTitle = "Selected Census Tracts",LegendTitle =
"Census Tracts",filterAddress = address,filterRadius = 1,alpha =
0.1,interactive = FALSE,geosObject = NULL,verbose = FALSE)

Create a map of an entire geography, using a variable.
mapper(mapDF = NULL,tableID = "B02001",variable = "B02001_002",variableSummary =
"B02001_001",geography = "county",year = 2022,state = 17,county
= 043,dispPerc = FALSE,MapTitle = "Selected Census Tracts",LegendTitle =
"Census Tracts",alpha = 0.1,interactive = FALSE,geosObject = NULL,
verbose = FALSE)

## End(Not run)
```

map_locations *Map Locations from Address List*

Description

This function takes a vector or dataframe containing addresses and geocodes them, and plots them.

Usage

```
map_locations(
  addressList,
  state = NULL,
  county = NULL,
  geosObject = NULL,
  verbose = FALSE
)
```

Arguments

addressList	Either a vector with addresses listed, or a dataframe needing formatting.
state	Parameter to filter by state.
county	Parameter to filter by county.
geosObject	Optional geosObject to speed up processing time.
verbose	Logical param to provide feedback.

Value

TMAP map plot

Examples

```
## Not run:
map_locations(addressList)
map_locations(addressList,state="IL",county=43)

## End(Not run)
```

profiler *ProfileR*

Description

A wrapper for capi() with additional parameters for profile creation.

Usage

```
profiler(
  name = NULL,
  year = NULL,
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
  censusVars = NULL,
  tableID = NULL,
  variables = NULL,
  geography = NULL,
  filterAddress = NULL,
  filterRadius = NULL,
  filterByGeoType = NULL,
  filterByGeoValue = NULL,
  filterSummary = FALSE,
  filterSummaryLevels = "root",
  state = NULL,
  county = NULL,
  tract = NULL,
  block_group = NULL,
  metro = NULL,
  ggr = NULL,
  geosObject = NULL,
  simpleReturn = FALSE,
  test = FALSE,
  fast = FALSE,
  verbose = FALSE,
  st = NULL
)
```

Arguments

name	User-supplied name for profile.
year	Year for data selection.
dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables
tableID	Specification for concept, or group: e.g., "B01001"
variables	A vector of all variables requested.
geography	Geography specification: e.g.: tract, county, state.
filterAddress	An address input used to generate a radius around, for filtering data.
filterRadius	A numeric value specifying the radius in miles around the address.
filterByGeoType	An irregular geo type to get a smaller overlapping set of tracts, block_groups or other geography from. Options are currently "metro", "place","combined_statistical_areas". E.g., Find all tracts in Chicago (place).

filterByGeoValue
: A value to find object for filtering. Either NAME or GEOID.

filterSummary
: Logical parameter to specify whether to filter out summary levels (typically _001 and therefore "root").

filterSummaryLevels
: Explicit description of lowest type denoting summary level. Also excludes lower levels.#' @param state Input (abb. or FIPS) of state for search.

state
: Input (abb. or FIPS) of state for search.

county
: Input (abb. or FIPS) of county for search.

tract
: Input (abb. or FIPS) of tract for search.

block_group
: Input (abb. or FIPS) of block group for search.

metro
: Input (abb. or FIPS) of metropolitan statistical area for search.

ggr
: Internal: to pass a get_geocode_radius() object to function.

geosObject
: Optional, attach geos object to simplify geo processes.

simpleReturn
: Param to return raw data, not formatted.

test
: Internal parameter for testing suite.

fast
: Internal parameter for pseudo_tableID and stat table (capi())

verbose
: Logical parameter to specify whether to produce verbose output.

st
: Internal parameter to provide timestamp consistency.

Value

data.frame

Examples

```
## Not run:
make_profile(x)

## End(Not run)
```

---

profile_helper *Profile Helper*

Description

A function designed to interactively create a vectorized list of selected variables to use for profiler() functionality.

Usage

```
profile_helper(
  tableID = NULL,
  year = NULL,
  allCols = FALSE,
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
```

```
  censusVars = NULL,
  test = FALSE,
  verbose = FALSE
)
```

Arguments

tableID	Vector with variables for inclusion. should be accessed.
year	Year of call.
allCols	Parameter to display all columns during review of variables for selection.
dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables
test	Internal: logical parameter to specificy testing environment.
verbose	Logical parameter to specify verbose output.

Value

Vector with variable values stored.

Examples

```
## Not run:
profile_builder(tableID=tableID,addNumbering=TRUE)

## End(Not run)
```

---

set_api_key *Set API Key*

Description

Sets the API key for the US Census API (see https://api.census.gov/data/key_signup.html) into the global environment for reuse with censusprofiler.

Usage

```
set_api_key(key = NULL, test = FALSE)
```

Arguments

key	The API key obtained for use with US Census Data API.
test	Internal, for testing purposes.

Value

A global environmental variable.

Examples

```
## Not run:
set_api_key(APIKEYGOESHERE)

## End(Not run)
```

`spatial_helper` *Spatial Helper*

Description

Internal helper function to create spatial objects by merging geoid with geo_tracts data.

Usage

```
spatial_helper(
  df,
  geography = "tract",
  state = NULL,
  county = NULL,
  geosObject = NULL,
  test = FALSE,
  verbose = FALSE
)
```

Arguments

`df`	Dataobject with GEOID column.
`geography`	Param specifying the geographical designation for census information.
`state`	Filter geo_var_builder by state to speed things up.
`county`	Filter geo_var_builder by county to speed things up.
`geosObject`	Optional geosObject to speed up processing time.
`test`	Internal: Logical param to bypass feedback and provide limited data for testing.
`verbose`	Logical param to provide feedback.

Value

An sf object

Examples

```
## Not run:
spatial_helper(data)

## End(Not run)
```

speedtest *Speedtest*

Description

Speedtest

Usage

```
speedtest(x, y)
```

Arguments

x	First comparison object
y	Second comparison object

Value

data.frame with time comparisons.

Examples

```
## Not run:
speedtest(x,y)

## End(Not run)
```

stat_helper *Utility: Stat Helper*

Description

A function to a) mutate a dataframe of profile variables with Z-scores, and/or b) to test whether individual entries are significant on Rmd reports.

Usage

```
stat_helper(
  data,
  statTable = NULL,
  variables = NULL,
  entryNum = 1,
  zThresh = 1.5,
  zType = "pct",
  tableID = NULL,
  typeFilter = NULL,
  variable = NULL,
  dataType = 4,
  year = 2022,
```

```
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
  censusVars = NULL,
  verbose = FALSE
)
```

Arguments

data	A profile data object created using create_profile().
statTable	A dataframe of census tracts or blocks created with stat_table_builder. Input can receive either object or file path.
variables	List of tableID names to test tableID input.
entryNum	For looping. If a single use, value is '1'. Otherwise, dynamically populate.
zThresh	The threshold for z-score divergence.
zType	Whether we're examining relative ('pct') or absolute ('estimate').
tableID	The tableID value for filtering.
typeFilter	Numeric value to filter type levels in analysis.
variable	The variable value for filtering.
dataType	Allows us to specify whether this is a df, dfCount, or dfNoSummary table.
year	Year for data call.
dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables
verbose	Set to 'TRUE' to print output on rmarkdown (or other) report.

Value

Either a mutated dataframe, or text output describing significqant z-score divergences.

Examples

```
## Not run:
stat_helper(df,entryNum=1,zThresh=1.5,zType="pct",tableID="B01001",dataType="df",verbose=TRUE)

## End(Not run)
```

---

stat_table_builder *Utility: Stat Table Builder*

Description

A utility function for generating a dataframe object of all census tracts in the United States, with variables from a variable list, Additionally, can calculate distribution statistics.

Usage

```
stat_table_builder(
  year = NULL,
  data = NULL,
  summary_table = FALSE,
  master_list = FALSE,
  compiler = FALSE,
  tableID = NULL,
  variables = NULL,
  geography = "tract",
  test = FALSE,
  geosObject = NULL,
  saveProgress = FALSE,
  stateStart = NULL,
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
  censusVars = NULL,
  verbose = FALSE
)
```

Arguments

year	Year value for variable selection.
data	A data object generated by "master_list" set to TRUE, to create summary_table.
summary_table	Set to 'TRUE' to create basic distribution statistics.
master_list	Set to 'TRUE' to download census tracts with variable list provided.
compiler	Set to 'TRUE' to stitch downloaded master list files into a single master file.
tableID	profileList object to pass to ACS call.
variables	Variable list (vector) object to pass to ACS call.
geography	Options to get "block group","tract","county" or "state".
test	Testing variable. Uses a limited data call for internal testing.
geosObject	Optional geosObject to speed up processing time.
saveProgress	Logical parameter to save individual downloaded files to /data/ folder to reduce runtime and memory load.
stateStart	Related to saveProgress, if the process is interrupted, a state FIPS code to begin the process at.
dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables
verbose	Logical parameter to specify whether to produce verbose output.

Value

A dataframe.

Examples

```
## Not run:
# Run summary statistics.
stat_table_builder(data=profile_all_summary_stats,
summary_table=TRUE)
# Download all census tracts.
stat_table_builder(master_list=TRUE)

## End(Not run)
```

---

tabler *TableR*

Description

A censusprofiler wrapper for flextables in RDF files.

Usage

```
tabler(
  data_object = NULL,
  mode = "summarize",
  tableID = NULL,
  variables = NULL,
  cols = c("labels", "estimate", "pct"),
  dispPerc = TRUE,
  type = NULL,
  shorten = NULL,
  sort = FALSE,
  sort_bygroup = FALSE,
  pctFilter = NULL,
  pdf = FALSE,
  usCompare = NULL,
  stateCompare = NULL,
  summaryLevels = 1,
  filterAddress = NULL,
  filterRadius = NULL,
  filterSummaryLevels = NULL,
  state = NULL,
  county = NULL,
  tract = NULL,
  block_group = NULL,
  geography = "tract",
  year = NULL,
  geosObject = NULL,
  dataset_main = "acs",
  dataset_sub = "acs5",
  dataset_last = NULL,
  censusVars = NULL,
  verbose = FALSE
)
```

Arguments

data_object	A census data object.
mode	Options for display calculations: "summarize", "summarizeinc","simple","bygeo","nosummary"
tableID	Formerly known as varBase, or concept, or group: i.e., "B01001"
variables	A vector of variables for the call.
cols	Defaults to "labels","estimate","pct", but can be modified.
dispPerc	Whether to display percentages. This may be set to FALSE when using median or mean data.
type	Parameter to filter by type ("root","summary","level_1", etc)
shorten	Numeric value specifying the size of a slice.
sort	Option to sort by value.
sort_bygroup	Logical flag to specify whether to sort by variable group.
pctFilter	A numeric value filtering the minimum percentage shown.
pdf	Set to TRUE for latex use.
usCompare	Pass comparison dataset offer a national compare value.
stateCompare	Pass comparison dataset offer a state-level compare value.
summaryLevels	How deep shaded levels will display.
filterAddress	Address of centroid to filter census tracts/blocks.
filterRadius	In miles, the radius for the filter.
filterSummaryLevels	Lowest summary level to include.
state	A state value to filter.
county	A county value to filter.
tract	Input (abb. or FIPS) of tract for search.
block_group	Input (abb. or FIPS) of block group for search.
geography	Geography designation for capi().
year	Year designation for capi().
geosObject	Optional geosObject object to speed up geo processing.
dataset_main	Selection parameters for get_census_variables (e.g. "acs")
dataset_sub	Selection parameters for get_census_variables (e.g. "acs5")
dataset_last	Selection parameters for get_census_variables (e.g. "cprofile")
censusVars	Passthrough object to bypass get_census_variables
verbose	Pass through param to create_profile() for feedback.

Value

A flextable object.

Examples

```
## Not run:
tabler(data_object=data,datatype="acs",dataset="acs5",mode="simple",
tableID="B02001",variables=NULL,summaryLevels=1,)

## End(Not run)
```

type_data *Type Data*

Description

Type Data

Usage

```
type_data(dataset, return = FALSE)
```

Arguments

dataset	A dataset to run type_data() on.
return	Whether to return the dataframe (TRUE) or type (FALSE)

Value

numeric value

Examples

```
## Not run:
type_data(x)

## End(Not run)
```

variable_builder *Variable Builder*

Description

A function to loop through and grab variables in a nested list for further analysis.

Usage

```
variable_builder(
  tableID = NULL,
  varStartNum = NULL,
  varEndNum = NULL,
  varSummaryNum = NULL,
  varArray = NULL,
  censusVars = NULL,
  verbose = FALSE
)
```

Arguments

tableID	Variable base, prior to the underscore.
varStartNum	Start number of variables.
varEndNum	End number of variables.
varSummaryNum	Summary number.
varArray	A manual list if need be.
censusVars	Reference to variable table.
verbose	Logical parameter to specify additional output.

Value

Returns either a vector, or a nested list.

Examples

```
## Not run:
variable_builder(c("B01001","B01002"),censusVars=CV.VARS,
varStartNum = c(1,1),varEndNum=c(3,3), varArray=NULL,varSummaryNum = c(1,1))

## End(Not run)
```