Package 'zippeR'

July 21, 2025

Title Working with United States ZIP Code and ZIP Code Tabulation Area Data

Version 0.1.2

Description Provides a set of functions for working with American postal codes, which are known as ZIP Codes. These include accessing ZIP Code to ZIP Code Tabulation Area (ZCTA) crosswalks, retrieving demographic data for ZCTAs, and tabulating demographic data for three-digit ZCTAs.

Depends R (>= 3.5)

License Apache License (>= 2)

URL https://github.com/pfizer-opensource/zippeR

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports cli, datasets, dplyr, httr, jsonlite, purrr, readr, sf, spatstat.univar, stats, stringr, tibble, tidycensus, tidyr, tigris

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Christopher Prener [aut, cre] (ORCID: <https://orcid.org/0000-0002-4310-9888>), Timothy Wiemken [aut] (ORCID: <https://orcid.org/0000-0002-8251-3007>), Angela Cook [aut]

Maintainer Christopher Prener <Christopher.Prener@pfizer.com>

Repository CRAN

Date/Publication 2025-04-25 20:50:02 UTC

Contents

`zi_aggregate` *Aggregate ZCTAs to Three-digit ZCTAs*

Description

This function takes input ZCTA data and aggregates it to three-digit areas, which are considerably larger. These regions are sometimes used in American health care contexts for publishing geographic identifiers.

Usage

```
zi_aggregate(.data, year, extensive = NULL, intensive = NULL,
    intensive_method = "mean", survey, output = "tidy", zcta = NULL,
    key = NULL)
```

Arguments

`.data`	A tidy set of demographic data containing one or more variables that should be aggregated to three-digit ZCTAs. This data frame or tibble should contain all five-digit ZCTAs within the three digit ZCTAs that you plan to use for aggregating data. See Details below for formatting requirements.
`year`	A four-digit numeric scalar for year. `zippeR` currently supports data for from 2010 to 2022. Different `survey` products are available for different years. See the `survey` parameter for more details.

`extensive` A character scalar or vector listing all extensive (i.e. count data) variables you wish to aggregate. These will be summed. For American Community Survey data, the margin of error will be calculated by taking the square root of the summed, squared margins of error for each five-digit ZCTA within a given three-digit ZCTA.

`intensive` A character scalar or vector listing all intensive (i.e. ratio, percent, or median data) variables you wish to aggregate. These will be combined using the approach listed for `intensive_method`.

`intensive_method` A character scalar; either `"mean"` (default) or `"median"`. In either case, a weighted approach is used where total population for each five-digit ZCTA is used to calculate individual ZCTAs' weights. For American Community Survey Data, this is applied to the margin of error as well.

`survey` A character scalar representing the Census product. It can be either a Decennial Census product (either `"sf1"` or `"sf3"`) or an American Community Survey product (either `"acs1"`, `"acs3"`, or `"acs5"`). For Decennial Census calls, only the 2010 Census is available. In addition, if a variable cannot be found in `"sf1"`, the function will look in `"sf3"`. Also note that `"acs3"` was discontinued after 2013.

`output` A character scalar; one of `"tidy"` (long output) or `"wide"` depending on the type of data format you want. If you are planning to join these data with geometric data, `"wide"` is the strongly encouraged format.

`zcta` An optional vector of ZCTAs that demographic data are requested for. If this is `NULL`, data will be returned for all ZCTAs. If a vector is supplied, only data for those requested ZCTAs will be returned. The vector can be created with `zi_get_geometry()`. If `style = "zcta5"`, this vector should be made up of five-digit `GEOID` values. If `style = "zcta3"`, this vector should be made up of three-digital `ZCTA3` values.

`key` A Census API key, which can be obtained at https://api.census.gov/data/key_signup.html. This can be omitted if `tidycensus::census_api_key()` has been used to write your key to your `.Renviron` file. You can check whether an API key has been written to `.Renviron` by using `Sys.getenv("CENSUS_API_KEY")`.

Value

A tibble containing all aggregated data requested in either `"tidy"` or `"wide"` format.

Examples

```
# load sample demographic data
mo22_demos <- zi_mo_pop

  # the above data can be replicated with the following code:
  # zi_get_demographics(year = 2022, variables = c("B01003_001", "B19013_001"),
  #   survey = "acs5")

# load sample geometric data
mo22_zcta3 <- zi_mo_zcta3
```

```
  # the above data can be replicated with the following code:
  # zi_get_geometry(year = 2022, style = "zcta3", state = "MO",
  #   method = "intersect")

# aggregate a single variable
zi_aggregate(mo22_demos, year = 2020, extensive = "B01003_001", survey = "acs5",
  zcta = mo22_zcta3$ZCTA3)


# aggregate multiple variables, outputting wide data
zi_aggregate(mo22_demos, year = 2020,
  extensive = "B01003_001", intensive = "B19013_001", survey = "acs5",
  zcta = mo22_zcta3$ZCTA3, output = "wide")
```

`zi_convert` *Convert Five-digit ZIP Codes to Three-digit ZIP Codes*

Description

This function converts five-digit ZIP Codes to three-digit ZIP Codes. The first three digits of a ZIP Code are known as the ZIP3 Code, and corresponds to the sectional center facility (SCF) that processes mail for a region.

Usage

```
zi_convert(.data, input_var, output_var)
```

Arguments

`.data`	A data frame containing a column of five-digit ZIP Codes.
`input_var`	A character scalar specifying the column name with the five-digit ZIP Codes in the data frame.
`output_var`	Optional; A character scalar specifying the column name to store the three-digit ZIP Codes in the data frame.

Value

A tibble containing the original data frame with a new column of three-digit ZIP Codes.

Examples

```
# add new column
## create sample data
df <- data.frame(id = c(1:3), zip5 = c("63005", "63139", "63636"))

## convert ZIP Codes to ZIP3, creating a new column
```

```
zi_convert(.data = df, input_var = zip5, output_var = zip3)

# overwrite existing column
## create sample data
df <- data.frame(id = c(1:3), zip = c("63005", "63139", "63636"))

## convert ZIP Codes to ZIP3, creating a new column
zi_convert(.data = df, input_var = zip)
```

`zi_crosswalk` *Crosswalk ZIP Codes with UDS, HUD, or a Custom Dictionary*

Description

This function compares input data containing ZIP Codes with a crosswalk file that will append ZCTAs. This is an important step because not all ZIP Codes have the same five digits as their enclosing ZCTA.

Usage

```
zi_crosswalk(.data, input_var, zip_source = "UDS", source_var,
    source_result, year = NULL, qtr = NULL, target = NULL, query = NULL,
    by = NULL, return_max = NULL, key = NULL, return = "id")
```

Arguments

`.data`	An "input object" that is data.frame or tibble that contains ZIP Codes to be crosswalked.
`input_var`	The column in the input data that contains five-digit ZIP Codes. If the input is numeric, it will be transformed to character data and leading zeros will be added.
`zip_source`	Required character scalar or data frame; specifies the source of ZIP Code crosswalk data. This can be one of either `"UDS"` (default) or `"HUD"`, or a data frame containing a custom dictionary.
`source_var`	Character scalar, required when `zip_source` is a data frame containing a custom dictionary; specifies the column name in the dictionary object that contains ZIP Codes.
`source_result`	Character scalar, required when `zip_source` is a data frame containing a custom dictionary; specifies the column name in the dictionary object that contains ZCTAs, GEOIDs, or other values.
`year`	Optional four-digit numeric scalar for year; varies based on source. For `"UDS"`, years 2009 through 2023 are available. For `"HUD"`, years 2010 through 2024 are available. Does not need to be specified when a custom dictionary is used.
`qtr`	Numeric scalar, required when `zip_code` is `"HUD"`. Integer value between 1 and 4, representing the quarter of the year.

target	Character scalar, required when zip_code is "HUD". Can be one of "TRACT", "COUNTY", "CBSA", "CBSADIV", "CD", and "COUNTYSUB".
query	Scalar or vector, required when zip_code is "HUD". This can be a five-digit numeric or character ZIP Code, a vector of ZIP Codes, a two-letter character state abbreviation, or "all".
by	Character scalar, required when zip_code is "HUD"; the column name to use for identifying the best match for a given ZIP Code. This could be either "residential", "commercial", or "total".
return_max	Logical scalar, required when zip_code is "HUD"; if TRUE (default), only the geography with the highest proportion of the ZIP Code type will be returned. If the ZIP Code straddles two states, two records will be returned. If FALSE, all records for the ZIP Code will be returned. Where a tie exists (i.e. two geographies each contain half of all addresses), the county with the lowest GEOID value will be returned.
key	Optional when zip_code is "HUD". This should be a character string containing your HUD API key. Alternatively, it can be stored in your .RProfile as hud_key.
return	Character scalar, specifies the type of output to return. Can be one of "id" (default), which appends only the crosswalked value, or "all", which returns the entire crosswalk file appended to the source data.

Value

A tibble with crosswalk values (or optionally, the full crosswalk file) appended based on the return argument.

Examples

```
# create sample data
df <- data.frame(id = c(1:3), zip5 = c("63005", "63139", "63636"))

# UDS crosswalk

  zi_crosswalk(df, input_var = zip5, zip_source = "UDS", year = 2022)


# HUD crosswalk
# you will need to replace INSERT_HUD_KEY with your own key
## Not run:
  zi_crosswalk(df, input_var = zip5, zip_source = "HUD", year = 2023,
    qtr = 1, target = "COUNTY", query = "MO", by = "residential",
    return_max = TRUE, key = INSERT_HUD_KEY)

## End(Not run)

# custom dictionary
## load sample crosswalk data to simulate custom dictionary
mo_xwalk <- zi_mo_hud
```

```
# prep crosswalk
# when a ZIP Code crosses county boundaries, the portion with the largest
# number of residential addresses will be returned
mo_xwalk <- zi_prep_hud(mo_xwalk, by = "residential", return_max = TRUE)

## crosswalk
zi_crosswalk(df, input_var = zip5, zip_source = mo_xwalk, source_var = zip5,
  source_result = geoid)
```

`zi_get_demographics` *Download Demographic Data for Five-digit ZCTAs*

Description

This function returns demographic data for five-digit ZIP Code Tabulation Areas (ZCTAs), which are rough approximations of many (but not all) USPS ZIP codes.

Usage

```
zi_get_demographics(year, variables = NULL, table = NULL,
    survey, output = "tidy", zcta = NULL, key = NULL)
```

Arguments

`year`	A four-digit numeric scalar for year. `zippeR` currently supports data for from 2010 to 2022. Different `survey` products are available for different years. See the `survey` parameter for more details
`variables`	A character scalar or vector of variable IDs.
`table`	A character scalar of a table ID (only one table may be requested per call).
`survey`	A character scalar representing the Census product. It can be either a Decennial Census product (either `"sf1"` or `"sf3"`) or an American Community Survey product (either `"acs1"`, `"acs3"`, or `"acs5"`). For Decennial Census calls, only the 2010 Census is available. In addition, if a variable cannot be found in `"sf1"`, the function will look in `"sf3"`. Also note that `"acs3"` was discontinued after 2013.
`output`	A character scalar; one of `"tidy"` (long output) or `"wide"` depending on the type of data format you want. If you are planning to pass these data to `zi_aggregate()`, you must choose `"tidy"`. If you are leaving these data as five-digit ZCTAs and are planning to join them with geometric data, `"wide"` is the strongly encouraged format.
`zcta`	An optional vector of ZCTAs that demographic data are requested for. If this is `NULL`, data will be returned for all ZCTAs. If a vector is supplied, only data for those requested ZCTAs will be returned. The vector can be created with `zi_get_geometry()` and should only contain five-digit ZCTAs.

key — A Census API key, which can be obtained at https://api.census.gov/data/key_signup.html. This can be omitted if tidycensus::census_api_key() has been used to write your key to your .Renviron file. You can check whether an API key has been written to .Renviron by using Sys.getenv("CENSUS_API_KEY").

Value

A tibble containing all demographic data requested in either "tidy" or "wide" format.

Examples

```
# download all ZCTAs
zi_get_demographics(year = 2012, variables = "B01003_001", survey = "acs5")

# limit output to subset of ZCTAs
## download all ZCTAs in Missouri, intersects method
mo20 <- zi_get_geometry(year = 2020, state = "MO", method = "intersect")

## download demographic data
zi_get_demographics(year = 2012, variables = "B01003_001", survey = "acs5",
    zcta = mo20$GEOID)
```

---

zi_get_geometry *Download and Optionally Geoprocess ZCTAs*

Description

This function returns geometric data for ZIP Code Tabulation Areas (ZCTAs), which are rough approximations of many (but not all) USPS ZIP codes. Downloading and processing these data will be heavily affected by your internet connection, your choice for the cb argument, and the processing power of your computer (if you select specific counties).

Usage

```
zi_get_geometry (year, style = "zcta5", return = "id", class = "sf",
    state = NULL, county = NULL, territory = NULL, cb = FALSE,
    starts_with = NULL, includes = NULL, excludes = NULL, method,
    shift_geo = FALSE)
```

Arguments

year — A four-digit numeric scalar for year. zippeR currently supports data between 2010 and 2023

style — A character scalar - either "zcta5" or "zcta3". See Details below.

`return` A character scalar; if `"id"` (default), only the five-digit number of each ZCTA (or three-digit if `style = "zcta3"`) is returned. This is the only valid option for `style = "zcta3"`. For `style = "zcta5"`, if `return = "full"`, all TIGER/Line columns are returned.

`class` A character scalar; if `"sf"` (default), a `sf` object suitable for mapping will be returned. If `"tibble"`, an object that omits the geometric data will be returned instead.

`state` A character scalar or vector with character state abbreviations (e.x. `"MO"`) or numeric FIPS codes (e.x. `29`). ZCTAs that are within the given states (determined based on a combination of `year` and `method`) will be returned. See Details below for more information. This argument is optional unless a argument is also specified for `county`.

`county` A character scalar or vector with character GEOIDs (e.x. `"29510"`). ZCTAs that are within the given states (determined based on a combination of `year` and `method`) will be returned. See Details below for more information. This argument is optional.

`territory` A character scalar or vector with character territory abbreviations (e.x. `"PR"`) or numeric FIPS codes (e.x. `72`). ZCTAs that are within the given territories will be returned. By default, all territories are excluded. The five territory abbreviations are: `"AS"` (American Samoa), `"GU"` (Guam), `"MP"` (Northern Mariana Islands), `"PR"` (Puerto Rico), and `"VI"` (U.S. Virgin Islands).

`cb` A logical scalar; if `FALSE`, the most detailed TIGER/Line data will be used for `style = "zcta5"`. If `TRUE`, a generalized (1:500k) version of the data will be used. The generalized data will download significantly faster, though they show less detail. According to the `tigris::zctas()` documentation, the download size if `TRUE` is ~65MB while it is ~500MB if `cb = FALSE`.

This argument does not apply to `style = "zcta3"`, which only returns generalized data. It also does not apply if `class = "tibble"`.

`starts_with` A character scalar or vector containing the first two digits of a GEOID or ZCTA3 value to return. It defaults to `NULL`, which will return all ZCTAs in the US. For example, supplying the argument `starts_with = c("63", "64")` will return only those ZCTAs or ZCTA3s that begin with 63 or 64. If you supply a state or a county, that will limit the data this argument is applied to, potentially leading to missed ZCTAs.

`includes` A character scalar or vector containing GEOID's or ZCTA3 values to include when finalizing output. This may be necessary depending on what is identified with the `method` argument.

`excludes` A character scalar or vector containing GEOID's or ZCTA3 values to exclude when finalizing output. This may be necessary depending on what is identified with the `method` argument.

`method` A character scalar - either `"intersect"` or `"centroid"`. See Details below.

`shift_geo` A logical scalar; if `TRUE`, Alaska, Hawaii, and Puerto Rico will be re-positioned so that the lie to the southwest of the continental United States. This defaults to `FALSE`, and can only be used when states are not listed for the `state` argument. It does not apply if `class = "tibble"`.

Details

This function contains options for both the type of ZCTA and, optionally, for how state and county data are identified. For type, either five-digit or three-digit ZCTA geometries are available. The three-digit ZCTAs were created by geoprocessing the five-digit boundaries for each year, and then applying a modest amount of simplification (with `sf::st_simplify()`) to reduce file size. The source files are available on GitHub at https://github.com/chris-prener/zcta3.

Since ZCTAs cross state lines, two methods are used to create these geometry data for years 2012 and beyond for states and all years for counties. The `"intersect"` method will return ZCTAs that border the states or counties selected. In most cases, this will result in more ZCTAs being returned than are actually within the states or counties selected. Conversely, the `"centroid"` method will return only ZCTAs whose centroids (geographical centers) lie within the states or counties named. In most cases, this will return fewer ZCTAs than actually lie within the states or counties selected. Users will need to review their data carefully and will likely need to use the `include` and `exclude` arguments to finalize the geographies returned.

For state-level data in 2010 and 2011, the Census Bureau published individual state files that will be utilized automatically by `zippeR`. If county-level data are requested for these years, the state-specific file will be used as a base before identifying ZCTAs within counties using either the `"intersect"` or `"centroid"` method described above.

Value

A `sf` object with ZCTAs matching the parameters specified above: either a nationwide file, a specific state or states, or a specific county or counties.

Examples

```
# five-digit ZCTAs
## download all ZCTAs for 2020 including territories
zi_get_geometry(year = 2020, territory = c("AS", "GU", "MP", "PR", "VI"),
    shift_geo = TRUE)

## download all ZCTAs for 2020 excluding territories
zi_get_geometry(year = 2020, shift_geo = TRUE)

## download all ZCTAs in a selection of states, intersects method
zi_get_geometry(year = 2020, state = c("IA", "IL", "MO"), method = "intersect")

## download all ZCTAs in a single county - St. Louis City, MO
zi_get_geometry(year = 2020, state = "MO", county = "29510",
    method = "intersect")

# three-digit ZCTAs
## download all ZCTAs for 2018 including territories
zi_get_geometry(year = 2018, territory = c("AS", "GU", "MP", "PR", "VI"),
    shift_geo = TRUE)
```

zi_label *Label ZIP Codes with Contextual Data*

Description

This function appends information about the city (for five-digit ZIP Codes) or area (for three-digit ZIP Codes) to a data frame containing these values. State is returned for both types of ZIP Codes. The function also optionally returns data on Sectional Center Facilities (SCFs) for three-digit ZIP Codes.

Usage

```
zi_label(.data, input_var, label_source = "UDS", source_var,
    type = "zip5", include_scf = FALSE, vintage = 2022)
```

Arguments

.data	An "input object" that is data.frame or tibble that contains ZIP Codes to be crosswalked.
input_var	The column in the input data that contains five-digit ZIP Codes. If the input is numeric, it will be transformed to character data and leading zeros will be added.
label_source	Required character scalar or data frame; specifies the source of the label data. This could be either 'UDS' (default) or 'USPS', or a data frame containing a custom dictionary.
source_var	Character scalar, required when label_source is a data frame containing a custom dictionary; specifies the column name in the dictionary object that contains ZIP Codes.
type	Character scalar, required when label_source is either label_source is 'UDS' or 'USPS'; one of either 'zip3' or 'zip5'. The 'zip3' type is only available from the 'USPS' source, while the 'zip5' type is available from 'UDS'.
include_scf	A logical scalar required when label_source = 'USPS' and type = 'zip3'; specifying whether to include the SCF (Sectional Center Facility) ID in the output. The default is FALSE.
vintage	Character or numeric scalar, required when label_source is either label_source is 'UDS' or 'USPS'; specifying the date for label_source = 'USPS' or the year of the data for label_source = 'UDS'. The zip_load_labels_list() function can be used to see available date values for label_source = 'USPS'.

Details

Labels are approximations of the actual location of a ZIP Code. For five-digit ZIP Codes, the city and state may or may not correspond to an individuals' mailing address city (since multiple cities may be accepted as valid by USPS for a particular ZIP Code) or state (since ZIP Codes may cross state lines).

For three-digit ZIP Codes, the area and state may or may not correspond to an individuals' mailing address state (since SCFs cover multiple states). For example, the three digit ZIP Code `010` covers Western Massachusetts in practice, but is assigned to the state of Connecticut.

Value

A tibble containing the original data with additional columns from the selected label data set appended.

Examples

```
# create sample data
df <- data.frame(
  id = c(1:3),
  zip5 = c("63005", "63139", "63636"),
  zip3 = c("630", "631", "636")
)

# UDS crosswalk

  zi_label(df, input_var = zip5, label_source = "UDS", vintage = 2022)


# USPS crosswalk

  zi_label(df, input_var = zip3, label_source = "USPS", type = "zip3",
    vintage = 202408)


# custom dictionary
## load sample ZIP3 label data to simulate custom dictionary
mo_label <- zi_mo_usps

## label
zi_label(df, input_var = zip3, label_source = mo_label, source_var = zip3,
  type = "zip3")
```

---

zi_list_zctas *List ZCTA GEOIDs for States*

Description

This function returns a vector of GEOIDs that represent ZCTAs in and around states, depending on the method selected. The two methods included described in Details below.

Usage

```
zi_list_zctas(year, state, method)
```

Arguments

year	A four-digit numeric scalar for year. zippeR currently supports data between 2010 and 2021.
state	A scalar or vector with state abbreviations (e.x. "MO") or FIPS codes (e.x. 29).
method	A character scalar - either "intersect" or "centroid". See Details below.

Details

Since ZCTAs cross state lines, two methods are used to create these vectors. The "intersect" method will return ZCTAs that border the state selected. In most cases, this will result in more ZCTAs being returned than are actually within the states(s) named in the state argument. Conversely, the "centroid" method will return only ZCTAs whose centroids (geographical centers) lie within the states named. In most cases, this will return fewer ZCTAs than actually lie within the state selected. Users will need to review their data carefully and, when using other zipperR functions, will likely need to use the include and exclude arguments to finalize the geographies returned.

Value

A vector of GEOIDs representing ZCTAs in and around the state selected.

Examples

```
# Missouri ZCTAs, intersect method
## return list
mo_zctas <- zi_list_zctas(year = 2021, state = "MO", method = "intersect")

## preview ZCTAs
mo_zctas[1:10]

# Missouri ZCTAs, centroid method
## return list
mo_zctas <- zi_list_zctas(year = 2021, state = "MO", method = "centroid")

## preview ZCTAs
mo_zctas[1:10]
```

---

zi_load_crosswalk *Load Crosswalk Files*

Description

Spatial data on USPS ZIP Codes are not published by the U.S. Postal Service or the U.S. Census Bureau. Instead, ZIP Codes can be converted to a variety of Census Bureau geographies using crosswalk files. This function reads in ZIP Code to ZIP Code Tabulation Area (ZCTA) crosswalk files from the former UDS Mapper project, which was sunset by the American Academy of Family Physicians in early 2024. It also provides access to the U.S. Department of Housing and Urban Development's ZIP Code crosswalk files, which provide similar functionality for converting ZIP Codes to a variety of geographies including counties.

Usage

```
zi_load_crosswalk(zip_source = "UDS", year, qtr = NULL, target = NULL,
    query = NULL, key = NULL)
```

Arguments

zip_source	Required character scalar; specifies the source of ZIP Code crosswalk data. This can be one of either "UDS" (default) or "HUD".
year	Required four-digit numeric scalar for year; varies based on source. For "UDS", years 2009 through 2023 are available. For "HUD", years 2010 through 2024 are available.
qtr	Numeric scalar, required when zip_code is "HUD". Integer value between 1 and 4, representing the quarter of the year.
target	Character scalar, required when zip_code is "HUD". Can be one of "TRACT", "COUNTY", "CBSA", "CBSADIV", "CD", and "COUNTYSUB".
query	Scalar or vector, required when zip_code is "HUD". This can be a five-digit numeric or character ZIP Code, a vector of ZIP Codes, a two-letter character state abbreviation, or "all".
key	Optional when zip_code is "HUD". This should be a character string containing your HUD API key. Alternatively, it can be stored in your .RProfile as hud_key.

Value

A tibble containing the crosswalk file.

Examples

```
 # former UDS mapper crosswalks
 zi_load_crosswalk(zip_source = "UDS", year = 2020)


## Not run:
 # HUD crosswalks
 # you will need to replace INSERT_HUD_KEY with your own key
 ## ZIP Code to CBSA crosswalk for all ZIP Codes
 zi_load_crosswalk(zip_source = "HUD", year = 2023, qtr = 1, target = "CBSA",
     query = "all", key = INSERT_HUD_KEY)

 ## ZIP Code to County crosswalk for all ZIP Codes in Missouri
 zi_load_crosswalk(zip_source = "HUD", year = 2023, qtr = 1, target = "COUNTY",
     query = "MO", key = INSERT_HUD_KEY)

 ## ZIP Code to Tract crosswalk for ZIP Code 63139 in St. Louis City
 zi_load_crosswalk(zip_source = "HUD", year = 2023, qtr = 1, target = "TRACT",
     query = 63139, key = INSERT_HUD_KEY)

## End(Not run)
```

`zi_load_labels` *Load Label Data*

Description

This function loads a specific label data set that can be used to label five or three-digit ZIP codes in a data frame.

Usage

```
zi_load_labels(source = "UDS", type = "zip5", include_scf = FALSE,
    vintage = 2022)
```

Arguments

`source`	A required character scalar; specifies the source of the label data. The only supported sources are `'UDS'` (default) and `'USPS'`.
`type`	A required character scalar; one of either `'zip3'` or `'zip5'`. The `'zip3'` type is only available from the `'USPS'` source, while the `'zip5'` type is available from `'UDS'`.
`include_scf`	A logical scalar required when `source = 'USPS'` and `type = 'zip3'`; specifying whether to include the SCF (Sectional Center Facility) ID in the output. The default is `FALSE`.
`vintage`	A required character or numeric scalar; specifying the date for `source = 'USPS'` or the year of the data for `source = 'UDS'`. The `zip_load_labels_list()` function can be used to see available date values for `source = 'USPS'`.

Details

Labels are approximations of the actual location of a ZIP Code. For five-digit ZIP Codes, the city and state may or may not correspond to an individuals' mailing address city (since multiple cities may be accepted as valid by USPS for a particular ZIP Code) or state (since ZIP Codes may cross state lines).

For three-digit ZIP Codes, the area and state may or may not correspond to an individuals' mailing address state (since SCFs cover multiple states). For example, the three digit ZIP Code `010` covers Western Massachusetts in practice, but is assigned to the state of Connecticut.

Value

A tibble with the specified label data for either five or three-digit ZIP Codes.

Examples

```
# zip5 labels via UDS
zi_load_labels(source = "UDS", type = "zip5", vintage = 2022)

# zip3 labels via USPS
```

```
zi_load_labels(source = "USPS", type = "zip3", vintage = 202408)
```

`zi_load_labels_list` *Load List of Available Label Data Sets*

Description

This function loads a list of available label data sets that can be used to label ZIP Codes. Currently, only three-digit ZIP Codes are supported.

Usage

```
zi_load_labels_list(type = "zip3")
```

Arguments

`type`	A character scalar specifying the type of label data to load. The only supported type is `'zip3'` (three-digit ZIP Codes).

Value

A tibble containing date values that can be used with `zi_load_labels`.

Examples

```
zi_load_labels_list(type = "zip3")
```

`zi_mo_hud` *Missouri HUD ZIP Code to County Crosswalk, 2023*

Description

A tibble containing the HUD ZIP Code to County Crosswalk file for Missouri's ZIP Codes in 2023's first quarter.

Usage

```
data(zi_mo_hud)
```

Format

A data frame with 1749 rows and 8 variables:

ZIP five-digit United States Postal Service ZIP Code

GEOID five-digit county FIPS code

RES_RATIO for ZIP Codes that cross county boundaries, the proportion of the ZIP Code's residential customers in the given county

BUS_RATIO for ZIP Codes that cross county boundaries, the proportion of the ZIP Code's commercial customers in the given county

OTH_RATIO for ZIP Codes that cross county boundaries, the proportion of the ZIP Code's other customers in the given county

TOT_RATIO for ZIP Codes that cross county boundaries, the proportion of the ZIP Code's total customers in the given county

CITY United States Postal Service city name

STATE United States Postal Service state abbreviation

Details

The data included in `zi_mo_hud` can be replicated with the following code: `zi_load_crosswalk(zip_source = "HUD", year = 2023, qtr = 1, target = "COUNTY", query = "MO")`. This assumes your HUD API key is stored in your `.Rprofile` file as `hud_key`.

Source

U.S. Department of Housing and Urban Development's ZIP Code crosswalk files

Examples

```
utils::str(zi_mo_hud)
utils::head(zi_mo_hud)
```

---

zi_mo_pop *Total Population and Median Household Income, Missouri ZCTAs 2022*

Description

A tibble containing the total population and median household income estimates from the 2018-2022 5-year U.S. Census Bureau American Communiy Survey estimates for Missouri five-digit ZIP Code Tabulation Areas (ZCTAs).

Usage

```
data(zi_mo_pop)
```

Format

A data frame with 2664 rows and 4 variables:

GEOID full GEOID string

variable variable, either `B01003_001` (total population) or `B19013_001` (median household income)

estimate value for associated `variable`

moe margin of error for associated `variable`

Details

The data included in `zi_mo_pop` can be replicated with the following code: `zi_get_demographics(year = 2022, variables = c("B01003_001", "B19013_001"), survey = "acs5")`.

Source

U.S. Census Bureau American Community Survey

Examples

```
utils::str(zi_mo_pop)
utils::head(zi_mo_pop)
```

`zi_mo_usps` *Missouri USPS Three-digit ZIP Code Labels, August 2024*

Description

A tibble containing the USPS Three-digit ZIP Code labels for August 2024.

Usage

```
data(zi_mo_usps)
```

Format

A data frame with 37 rows and 3 variables:

zip3 three-digit United States Postal Service ZIP Code

label_area area associated with the three-digit ZIP Code

label_state state associated with the three-digit ZIP Code

Details

The data included in `zi_mo_usps` can be replicated with the following code: `zi_load_labels(type = "zip3", source = "USPS", vintage = 202408)`. After downloading the data, subset to `label_state == "MO"`.

Source

U.S. Postal Service Facility Access and Shipment Tracking (FAST) Database

Examples

```
utils::str(zi_mo_usps)
utils::head(zi_mo_usps)
```

---

zi_mo_zcta3 *Missouri Three-digit ZCTAs, 2022*

Description

A simple features data set containing the geometric data for Missouri's three-digit ZIP Code Tabulation Areas (ZCTAs) for 2022, derived from the U.S. Census Bureau's 2022 TIGER/Line shapefiles.

Usage

```
data(zi_mo_zcta3)
```

Format

A data frame with 31 rows and 2 variables:

ZCTA3 three-digit ZCTA value

geometry simple features geometry

Details

The data included in `zi_mo_zcta3` can be replicated with the following code: `zi_get_geometry(year = 2022, style = "zcta3", state = "MO", method = "intersect")`.

Source

U.S. Census Bureau's TIGER/Line database

Examples

```
utils::str(zi_mo_zcta3)
utils::head(zi_mo_zcta3)
```

zi_prep_hud *Convert HUD Crosswalk Data to Finalized Crosswalk*

Description

The output from `zi_load_crosswalk()` for HUD data requires additional processing to be used in the `zi_crosswalk()` function. This function prepares the HUD data for use in joins.

Usage

```
zi_prep_hud(.data, by, return_max = TRUE)
```

Arguments

`.data`	The output from `zi_load_crosswalk()` with HUD data.
`by`	Character scalar; the column name to use for identifying the best match for a given ZIP Code. This could be either `"residential"`, `"commercial"`, or `"total"`.
`return_max`	Logical scalar; if `TRUE` (default), only the county with the highest proportion of the ZIP Code type will be returned. If the ZIP Code straddles two states, two records will be returned. If `FALSE`, all records for the ZIP Code will be returned. Where a tie exists (i.e. two counties each contain half of all addresses), the county with the lowest `GEOID` value will be returned.

Value

A tibble that has been further prepared for use as a crosswalk.

Examples

```
# load sample crosswalk data
mo_xwalk <- zi_mo_hud

  # the above data can be replicated with the following code:
  # zi_load_crosswalk(zip_source = "HUD", year = 2023, qtr = 1,
  #   target = "COUNTY", query = "MO")

# prep crosswalk
# when a ZIP Code crosses county boundaries, the portion with the largest
# number of residential addresses will be returned
zi_prep_hud(mo_xwalk, by = "residential", return_max = TRUE)
```

`zi_repair` *Repair ZIP Code or ZCTA Vector*

Description

This function repairs two of the four conditions identified in the validation checks with `zi_validate()`. For the other two conditions, values are conveted `NA`. See Details below for the specific changes made.

Usage

```
zi_repair(x, style = "zcta5")
```

Arguments

`x`	A vector containing ZIP or ZCTA values to be repaired.
`style`	A character scalar - either `"zcta5"` or `"zcta3"`.

Details

The `zi_repair()` function addresses four conditions:

- If the input vector is numeric, it will be converted to character data.
- If there are values less than five characters (if `style = "zcta5"`, the default), or three characters (if `style = "zcta3"`), they will be padded with leading zeros.
- If there are input values over five characters (if `style = "zcta5"`, the default), or three characters (if `style = "zcta3"`), they will be converted to `NA`.
- If there are input values that have non-numeric characters, they will be converted to `NA`.

Since two of the four steps will result in `NA` values, it is strongly recommended to attempt to manually fix these issues first.

Value

A repaired vector of ZIP or ZCTA values.

Examples

```
# sample five-digit ZIPs with character
zips <- c("63088", "63108", "zip")

# failed validation
zi_validate(zips)

# repair
zips <- zi_repair(zips)

# successful validation
```

```
zi_validate(zips)
```

`zi_validate` *Validate ZIP Code or ZCTA Vector*

Description

This function validates vectors of ZIP Code or ZCTA values. It is used internally throughout `zippeR` for data validation, but is exported to facilitate troubleshooting.

Usage

```
zi_validate(x, style = "zcta5", verbose = FALSE)
```

Arguments

`x`	A vector containing ZIP or ZCTA values to be validated.
`style`	A character scalar - either `"zcta5"` (default) or `"zcta3"`.
`verbose`	A logical scalar; if `FALSE` (default), an overall evaluation will be returned. If `TRUE`, a `tibble` object listing validation criteria and results will be returned.

Details

The `zi_validate()` function checks for four conditions:

- Is the input vector character data? This is important because of USPS's use of leading zeros in ZIP codes and ZCTAs.
- Are all values five characters (if `style = "zcta5"`, the default), or three characters (if `style = "zcta3"`)?
- Are any input values over five characters (if `style = "zcta5"`, the default), or three characters (if `style = "zcta3"`)?
- Do any input values have non-numeric characters?

The questions provide a basis for repairing issues identified with `zi_repair()`.

Value

Either a logical value (if `verbose = FALSE`) or a `tibble` containing validation criteria and results.

Examples

```
# sample five-digit ZIPs
zips <- c("63088", "63108", "63139")

# successful validation
zi_validate(zips)

# sample five-digit ZIPs in data frame
zips <- data.frame(id = c(1:3), ZIP = c("63139", "63108", "00501"), stringsAsFactors = FALSE)

# successful validation
zi_validate(zips$ZIP)

# sample five-digit ZIPs with character
zips <- c("63088", "63108", "zip")

# failed validation
zi_validate(zips)
zi_validate(zips, verbose = TRUE)
```

Index