# Package 'censusxy'

February 18, 2021

**Title** Access the U.S. Census Bureau's Geocoding A.P.I. System

**Version** 1.0.1

**Description** Provides access to the U.S. Census Bureau's A.P.I for matching American street addresses with their longitude and latitude. This includes both single address matching as well as batch functionality for multiple addresses. Census geographies can be appended to addresses if desired, and reverse geocoding of point locations to census geographies is also supported.

**Depends** R (>= 3.3)

**License** GPL-3

**URL** https://github.com/slu-openGIS/censusxy

**BugReports** https://github.com/slu-openGIS/censusxy/issues

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**Imports** httr

**Suggests** covr, knitr, parallel, rmarkdown, sf, testthat

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Christopher Prener [aut, cre] (<https://orcid.org/0000-0002-4310-9888>), Branson Fox [aut] (<https://orcid.org/0000-0002-4361-2811>)

**Maintainer** Christopher Prener <chris.prener@slu.edu>

**Repository** CRAN

**Date/Publication** 2021-02-18 06:30:02 UTC

## R topics documented:

---

`cxy_benchmarks` *Get Current Valid Benchmarks*

---

## Description

Get Current Valid Benchmarks

## Usage

```
cxy_benchmarks()
```

## Value

A data.frame containing valid Census Benchmarks

## Examples

```
cxy_benchmarks()
```

---

`cxy_geocode` *Batch Geocode Parsed Addresses*

---

## Description

Provides access to the US Census Bureau batch endpoints for locations and geographies. The function implements iteration and optional parallelization in order to geocode datasets larger than the API limit of 10,000 and more efficiently than sending 10,000 per request. It also supports multiple outputs, including SF class objects.

## Usage

```
cxy_geocode(
  .data,
  id = NULL,
  street,
  city = NULL,
  state = NULL,
  zip = NULL,
```

```
  return = "locations",
  benchmark = "Public_AR_Current",
  vintage = NULL,
  timeout = 30,
  parallel = 1,
  class = "dataframe",
  output = "simple"
)
```

## Arguments

| | |
|---|---|
| .data | data.frame containing columns with structured address data |
| id | Optional String - Name of column containing unique ID |
| street | String - Name of column containing street address |
| city | Optional String - Name of column containing city |
| state | Optional String - Name of column containing state |
| zip | Optional String - Name of column containing zip code |
| return | One of 'locations' or 'geographies' denoting returned information from the API |
| benchmark | Optional Census Benchmark to geocode against. See Details. |
| vintage | Optional Census Vintage to geocode against. See Details. |
| timeout | Numeric, in minutes, how long until request times out |
| parallel | Integer, number of cores greater than one if parallel requests are desired. See Details. |
| class | One of 'dataframe' or 'sf' denoting the output class. 'sf' will only return matched addresses. |
| output | One of 'simple' or 'full' denoting the returned columns. Simple returns just coordinates. |

## Details

Parallel requests are not currently supported on Windows. You may not specify more cores than the system reports are available If you do, the maximum number of available cores will be used.

To obtain current valid benchmarks, use the `cxy_benchmarks()` function

If you want to append census geographies, you must specify a valid vintage for your benchmark. You may use the `cxy_vintages()` function to obtain valid Vintages. See `vignette('censusxy')` for a full walkthrough.

## Value

A data.frame or sf object containing geocoded results

## Examples

```
# load data
x <- stl_homicides[1:10,]

# geocode
cxy_geocode(x, street = 'street_address', city = 'city', state = 'state', zip = 'postal_code',
   return = 'locations', class = 'dataframe', output = 'simple')
```

---

`cxy_geography` *Geocode Single Coordinate Pair*

---

## Description

Provides access to the GeoLookup API of the US Census Bureau. Returns census geographies for a single geographic point.

## Usage

```
cxy_geography(
  lon,
  lat,
  benchmark = "Public_AR_Current",
  vintage = "Current_Current"
)
```

## Arguments

| | |
|---|---|
| `lon` | Numeric or String Containing Longitude (x) of Point |
| `lat` | Numeric or String Containing Latitude (y) of Point |
| `benchmark` | Optional ID or Name of Census Benchmark. See Details. |
| `vintage` | Optional ID or Name of Census Vintage. See Details. |

## Details

This function can be used to locate geographic information given a geographic point. It does not provide an address like a reverse-geocoder

To obtain current valid benchmarks, use the `cxy_benchmarks()` function

To use this function, you must specify a valid vintage for your benchmark. You may use the `cxy_vintages()` function to obtain valid Vintages for a given benchmark. See `vignette('censusxy')` for a full walkthrough.

## Value

A data.frame containing matched address or NULL if not matches

## Examples

```
cxy_geography(lon = -90.23324, lat = 38.63593)
```

---

`cxy_oneline` *Geocode Single One Line Address*

---

## Description

Provides access to the oneline single address geocoding API from the US Census Bureau. This can be used with an address that is not parsed.

## Usage

```
cxy_oneline(
  address,
  return = "locations",
  benchmark = "Public_AR_Current",
  vintage = NULL
)
```

## Arguments

| | |
|---|---|
| `address` | String containing a single line address |
| `return` | One of 'locations' or 'geographies' See Details. |
| `benchmark` | Optional ID or Name of Census Benchmark. See Details. |
| `vintage` | Optional ID or Name of Census Vintage. See Details. |

## Details

To obtain current valid benchmarks, use the `cxy_benchmarks()` function.

If you want to append census geographies, you must specify a valid vintage for your benchmark. You may use the `cxy_vintages()` function to obtain valid Vintages. See `vignette('censusxy')` for a full walkthrough.

## Value

A data.frame containing matched address or NULL if not matches

## Examples

```
cxy_oneline(address = "20 N Grand Blvd, St Louis, MO 63108", return = "locations")
```

cxy_single *Geocode Single Parsed Address*

## Description

Provides access to the structured single address geocoding API from the US Census Bureau.

## Usage

```
cxy_single(
  street,
  city = NULL,
  state = NULL,
  zip = NULL,
  return = "locations",
  benchmark = "Public_AR_Current",
  vintage = NULL
)
```

## Arguments

| | |
|---|---|
| street | String containing street address |
| city | Optional String containing city |
| state | Optional String containing state |
| zip | Optional String or Integer containing 5-digit Zip Code |
| return | One of 'locations' or 'geographies' See Details. |
| benchmark | Optional ID or Name of Census Benchmark. See Details. |
| vintage | Optional ID or Name of Census Vintage. See Details. |

## Details

To obtain current valid benchmarks, use the `cxy_benchmarks()` function.

If you want to append census geographies, you must specify a valid vintage for your benchmark. You may use the `cxy_vintages()` function to obtain valid Vintages. See `vignette('censusxy')` for a full walkthrough.

## Value

A data.frame containing matched address or NULL if not matches

## Examples

```
cxy_single(street = "20 N Grand Blvd", city = "St Louis", state = "MO", zip = "63108",
    return = "locations")
```

## cxy_vintages *Get Current Valid Vintages*

### Description

Get Current Valid Vintages

### Usage

```
cxy_vintages(benchmark)
```

### Arguments

| | |
|---|---|
| benchmark | Name or ID of Census Benchmark |

### Value

A data.frame containing valid Census Vintages for a given benchmark

### Examples

```
cxy_vintages('Public_AR_Current')
```

## stl_homicides *Homicides in the City of St. Louis, 2008 - 2018*

### Description

An example data set containing the addresses for homicides reported by the Saint Louis Metropolitan Police Department

### Usage

```
data(stl_homicides)
```

### Format

A tibble with 1822 rows and 6 variables:

**street_address** number, street and street suffix where homicide occured

**year** year homicide occurred

**date** data homicide occurred

**state** state abbreviation of location, in these data, all "MO"

**postal_code** zipcode/postal code of location, in these data all NA

**city** city of location, in these data all "St. Louis"

## Source

St. Louis Metropolitan Police Department

## Examples

```
str(stl_homicides)
head(stl_homicides)
```

---

`stl_homicides_small` *Homicides in the City of St. Louis July, 2018*

---

## Description

An example data set containing the addresses for homicides reported by the Saint Louis Metropolitan Police Department

## Usage

```
data(stl_homicides_small)
```

## Format

A tibble with 24 rows and 6 variables:

**street_address** number, street and street suffix where homicide occured

**year** year homicide occurred

**date** data homicide occurred

**state** state abbreviation of location, in these data, all "MO"

**postal_code** zipcode/postal code of location, in these data all NA

**city** city of location, in these data all "St. Louis"

## Source

St. Louis Metropolitan Police Department

## Examples

```
str(stl_homicides_small)
head(stl_homicides_small)
```

# Index