Package 'tidygeocoder'

July 15, 2020

Type Package

Title Geocoding Made Easy

Version 1.0.0

Description An intuitive interface for getting data from geocoder services.

URL https://jessecambon.github.io/tidygeocoder,

https://github.com/jessecambon/tidygeocoder

BugReports https://github.com/jessecambon/tidygeocoder/issues

Depends R (>= 3.20)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports tibble, dplyr, httr, jsonlite

RoxygenNote 7.1.0

Suggests knitr, DT, rmarkdown, ggplot2, ggrepel, maps, testthat (>= 2.1.0)

VignetteBuilder knitr

NeedsCompilation no

Author Jesse Cambon [aut, cre]

Maintainer Jesse Cambon <jesse.cambon@gmail.com>

Repository CRAN

Date/Publication 2020-07-15 13:40:02 UTC

R topics documented:

api_parameter_reference
                          Geocoder Service API Parameter Reference

---

Description

This dataset contains the mapping that allows this package to use a universal syntax to specify parameters for different geocoder services.

The "generic_name" field is a universal field name while the api_name field shows the specific parameter name for the given geocoder service (method). When the api_name is missing this means that the parameter is not supported by the given geocoder service. When generic_name is missing this means the parameter is specific to that geocoding service.

Reference the documentation for geo for more information. Also reference vignette("tidygeocoder") for more details on constructing API queries.

Usage

    api_parameter_reference

Format

A tibble dataframe

method Geocoder service name

generic_name Universal parameter name

api_name Name of the parameter for the specified geocoder service

default_value Default value of the parameter

required Is the parameter required by the specified geocoder service?

Source

Links to API documentation for each geocoder service are below.

- Census
- Nominatim ("osm")
- Geocodio
- Location IQ ("iq")

See Also

[get_api_query](#) [query_api](#)

---

extract_results	*Extract Geocoder Results*

Description

Parses the output of the [query_api](#) function. Latitude and longitude are extracted into the first two columns of the returned dataframe. This function is not used for batch geocoded results.

Usage

```
extract_results(method, response, full_results = TRUE, flatten = TRUE)
```

Arguments

method	method name
response	content from the geocoder service (returned by the [query_api](#) function)
full_results	if TRUE then the full results (not just latitude and longitude) will be returned.
flatten	if TRUE then flatten any nested dataframe content

Value

geocoder results in tibble format

See Also

[get_api_query](#) [query_api](#) [geo](#)

---

geo	*Geocode addresses*

Description

Geocodes addresses given as character values. The [geocode](#) function utilizes this function on addresses contained in dataframes. See example usage in vignette("tidygeocoder")

Note that not all geocoder services support certain address component parameters. For example, the Census geocoder only covers the United States and does not have a "country" parameter. Refer to [api_parameter_reference](#) for more details on geocoder services and API usage.

This function uses the [get_api_query](#), [query_api](#), and [extract_results](#) functions to create, execute, and parse the geocoder API queries.

Usage

```
geo(
  address = NULL,
  street = NULL,
  city = NULL,
  county = NULL,
  state = NULL,
  postalcode = NULL,
  country = NULL,
  method = "census",
  cascade_order = c("census", "osm"),
  lat = lat,
  long = long,
  limit = 1,
  min_time = NULL,
  api_url = NULL,
  timeout = 20,
  mode = "",
  full_results = FALSE,
  unique_only = FALSE,
  return_addresses = TRUE,
  flatten = TRUE,
  batch_limit = 10000,
  verbose = FALSE,
  no_query = FALSE,
  custom_query = list(),
  return_type = "locations",
  iq_region = "us",
  geocodio_v = 1.6
)
```

Arguments

address	single line address (ie. '1600 Pennsylvania Ave NW, Washington, DC'). Do not combine with the address component arguments below (street, city , county, state, postalcode, country).
street	street address (ie. '1600 Pennsylvania Ave NW')
city	city (ie. 'Tokyo')
county	county (ie. 'Jefferson')
state	state (ie. 'Kentucky')
postalcode	postalcode (zip code if in the United States)
country	country (ie. 'Japan')
method	the geocoder service to be used. Refer to 'api_parameter_reference' and the API documentation for each geocoder service for usage details and limitations.

- "census": US Census Geocoder. US street-level addresses only. Can perform batch geocoding.

- "osm": Nominatim (OSM). Worldwide coverage.
- "geocodio": Commercial geocoder. Covers US and Canada and has batch geocoding capabilities. Requires an API Key to be stored in the "GEOCODIO_API_KEY" environmental variable.
- "iq": Commercial Nominatim geocoder service. Requires an API Key to be stored in the "LOCATIONIQ_API_KEY" environmental variable.
- "cascade" : Attempts to use one geocoder service and then uses a second geocoder service if the first service didn't return results. The services and order is specified by the cascade_order argument. Note that this is not compatible with full_results = TRUE as geocoder services have different columns that they return.

cascade_order	a vector with two character values for the method argument in the order in which the geocoder services will be attempted for method = "cascade"
lat	latitude column name. Can be quoted or unquoted (ie. lat or 'lat').
long	longitude column name. Can be quoted or unquoted (ie. long or 'long').
limit	number of results to return per address. Note that limit > 1 is not compatible with batch geocoding if return_addresses = TRUE.
min_time	minimum amount of time for a query to take (in seconds) if using Location IQ or OSM. This parameter is used to abide by API usage limits. You can set it to a lower value (ie. 0) if using a local Nominatim server, for instance.
api_url	Custom API URL. If specified, the default API URL will be overridden. This can be used to specify a local Nominatim server.
timeout	query timeout (in minutes)
mode	set to 'batch' to force batch geocoding and 'single' to force single address geocoding (one address per query). If not specified then batch geocoding will be used if available (given method selected) when multiple addresses are provided, otherwise single address geocoding will be used.
full_results	returns all data from geocoder service if TRUE
unique_only	only return results for unique addresses if TRUE
return_addresses	
	return input addresses with results if TRUE
flatten	if TRUE then any nested dataframes in results are flattened
batch_limit	limit to the number of addresses in a batch geocoding query. Both geocodio and census batch geocoders have a 10,000 address limit so this is the default.
verbose	if TRUE then detailed logs are output to the console
no_query	if TRUE then no queries are sent to the geocoder and verbose is set to TRUE
custom_query	API-specific parameters to be used, passed as a named list (ie. 'list(vintage = 'Current_Census2010')')
return_type	(census only) 'locations' (default) or 'geographies' which returns additional census geography columns. See the Census geocoder API documentation for more details.
iq_region	'us' (default) or 'eu'. Used for establishing API URL for the 'iq' method
geocodio_v	version of geocodio api. 1.6 is default. Used for establishing API URL for the 'geocodio' method.

Value

parsed results from the geocoder service

See Also

[api_parameter_reference](#)

Examples

```
geo(street = "600 Peachtree Street NE", city = "Atlanta",
 state = "Georgia", method = "census")

geo(address = c("Tokyo, Japan", "Lima, Peru", "Nairobi, Kenya"),
 method = 'osm')

geo(county = 'Jefferson', state = "Kentucky", country = "US",
     method = 'osm')
```

---

geocode   *Geocode addresses in a dataframe*

Description

Takes a dataframe containing addresses as an input and returns the dataframe results from a specified geocoder service by using the [geo](#) function. See example usage in vignette("tidygeocoder").

This function passes all additional parameters (...) to the [geo](#) function, so you can refer to its documentation for more details on possible arguments.

Note that the arguments used for specifying address columns (address, street, city, county, state, postalcode, and country) accept either quoted or unquoted column names (ie. "address_col" and address_col are both acceptable).

Usage

```
geocode(
  .tbl,
  address = NULL,
  street = NULL,
  city = NULL,
  county = NULL,
  state = NULL,
  postalcode = NULL,
  country = NULL,
  lat = lat,
```

```
    long = long,
    return_addresses = FALSE,
    unique_only = FALSE,
    ...
)
```

Arguments

`.tbl`	dataframe containing addresses
`address`	single line street address column name. Do not combine with address component arguments (street, city, county, state, postalcode, country)
`street`	street address column name
`city`	city column name
`county`	county column name
`state`	state column name
`postalcode`	postalcode column name (zip code if in the United States)
`country`	country column name
`lat`	latitude column name. Can be quoted or unquoted (ie. lat or 'lat').
`long`	longitude column name. Can be quoted or unquoted (ie. long or 'long').
`return_addresses`	
	if TRUE then addresses with standard names will be returned This is defaulted to FALSE because the address fields are already in the input dataset
`unique_only`	if TRUE then only unique addresses and results will be returned. The input dataframe's format is not preserved. Addresses will also be returned if TRUE (overrides return_addresses argument).
`...`	arguments passed to the [geo](#) function

Value

input dataframe (.tbl) with geocoder results appended as columns

See Also

[geo](#)

Examples

```
library(dplyr)
sample_addresses[1:3,] %>% geocode(addr)

louisville[1:2,] %>% geocode(street = street, city = city, state = state,
  postalcode = zip)

sample_addresses[8:9,] %>% geocode(addr, method = 'osm',
  lat = 'lattes', long = 'longos')
```

```
sample_addresses[1:3,] %>% geocode(addr, method = 'cascade',
  lat = latitude, long = longitude)
```

---

geo_cascade   *Convenience function for calling the* geo *function with method = 'cas-*
                                cade'

Description

Convenience function for calling the geo function with method = 'cascade'

Usage

```
geo_cascade(..., cascade_order = c("census", "osm"))
```

Arguments

...	arguments passed from and to the geo function
cascade_order	a vector with two character values for the method argument in the order in which the geocoder services will be attempted

---

geo_census   *Convenience functions for calling the* geo *function with a specified*
                                method.

Description

Convenience functions for calling the geo function with a specified method.

Usage

```
geo_census(...)

geo_osm(...)

geo_geocodio(...)

geo_iq(...)
```

Arguments

...	arguments to be passed to the geo function

---

get_api_query   *Construct a Geocoder API Query*

Description

The geocoder API query is created using universal "generic" parameters and optional api-specific "custom" parameters. Generic parameters are converted into api parameters using the api_parameter_reference dataset.

The query_api function executes the queries created by this function.

Usage

```
get_api_query(method, generic_parameters = list(), custom_parameters = list())
```

Arguments

method   method name (ie. 'census')

generic_parameters
                universal 'generic' parameters

custom_parameters
                custom api-specific parameters

Value

named list of parameters

See Also

query_api geo

---

louisville   *Louisville, Kentucky Street Addresses*

Description

Louisville, Kentucky Street Addresses

Usage

```
louisville
```

Format

A tibble dataframe with component street addresses

street Description of the address

city Single line address

state state

zip zip code

Source

---

query_api   *Execute a Geocoder API Query*

Description

The [`get_api_query`](#) function can create queries for this function to execute.

Usage

```
query_api(
  api_url,
  query_parameters,
  mode = "single",
  batch_file = NULL,
  address_list = NULL,
  content_encoding = "UTF-8",
  timeout = 20
)
```

Arguments

api_url	Base URL of the API. query parameters are appended to this
query_parameters	
	api query parameters in the form of a named list
mode	"single" : geocode a single address (all methods) "list" : batch geocode a list of addresses (geocodio) "file" : batch geocode a file of addresses (census)
batch_file	a csv file of addresses to upload (census)
address_list	a list of addresses for batch geocoding (geocodio) should be 'json' for geocodio and 'multipart' for census
content_encoding	
	Encoding to be used for parsing content. "UTF-8" is the default, but "ISO-8859-1" is used for Census batch queries.
timeout	timeout in minutes

Value

raw results from the query

See Also

[get_api_query](#) [extract_results](#) [geo](#)

---

sample_addresses *Some sample addresses for testing*

Description

Some sample addresses for testing

Usage

sample_addresses

Format

A tibble dataframe with single line addresses

name Description of the address

addr Single line address

Index