

Package ‘nominatimlite’

May 9, 2026

Type Package

Title Interface with 'Nominatim' API Service

Version 0.5.0

Description Lite interface for getting data from 'OSM' service
'Nominatim' <<https://nominatim.org/release-docs/latest/>>. Extract coordinates from addresses, find places near a set of coordinates and return spatial objects on 'sf' format.

License MIT + file LICENSE

URL <https://dieghernan.github.io/nominatimlite/>,
<https://github.com/dieghernan/nominatimlite>

BugReports <https://github.com/dieghernan/nominatimlite/issues>

Depends R (>= 3.6.0)

Imports dplyr (>= 1.0.0), jsonlite (>= 1.7.0), sf (>= 0.9.0), testthat (>= 3.0.0), utils

Suggests arcgeocoder, ggplot2 (>= 3.0.0), knitr, lifecycle, quarto, rmarkdown, tibble (>= 3.0.0), tidygeocoder

VignetteBuilder quarto

Config/Needs/website dieghernan/gitdevr, pak, tidyverse, leaflet, reactable, crosstalk, tidyr, htmltools

Config/testthat/edition 3

Config/testthat/parallel false

Copyright Data © OpenStreetMap contributors, ODbL 1.0.
<<https://www.openstreetmap.org/copyright>>

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

X-schema.org-applicationCategory cartography

X-schema.org-keywords r, geocoding, openstreetmap, address, nominatim, reverse-geocoding, rstats, shapefile, r-package, spatial, cran, api-wrapper, api, gis

NeedsCompilation no

Author Diego Hernangómez [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0001-8457-4658>>),
Jindra Lacko [ctb, rev] (ORCID:
<<https://orcid.org/0000-0002-0375-5156>>),
Alex White [ctb],
OpenStreetMap [cph] (For the data)

Maintainer Diego Hernangómez <diego.hernangomezherrero@gmail.com>

Repository CRAN

Date/Publication 2026-03-18 06:12:39 UTC

Contents

bbox_to_poly	2
geo_address_lookup	4
geo_address_lookup_sf	5
geo_amenity	7
geo_amenity_sf	10
geo_lite	12
geo_lite_sf	14
geo_lite_struct	16
geo_lite_struct_sf	18
osm_amenities	20
reverse_geo_lite	21
reverse_geo_lite_sf	23

Index 26

bbox_to_poly	<i>Coerce a bounding box to a sfc POLYGON object</i>
--------------	--

Description

Create a [sfc](#) object from the coordinates of a bounding box.

Usage

```
bbox_to_poly(bbox = NA, xmin = NA, ymin = NA, xmax = NA, ymax = NA, crs = 4326)
```

Arguments

bbox	Numeric vector of 4 elements representing the coordinates of the bounding box. Values should be <code>c(xmin, ymin, xmax, ymax)</code> .
xmin, ymin, xmax, ymax	Alternatively, you can use these named parameters instead of <code>bbox</code> .
crs	coordinate reference system, something suitable as input to st_crs

Details

Bounding boxes can be located using online tools such as <https://boundingbox.klokantech.com/>.

Value

A `sfc` object of class POLYGON.

See Also

`sf::st_as_sfc()` and `sf::st_sfc()`.

Get `sf` objects: `geo_address_lookup_sf()`, `geo_amenity_sf()`, `geo_lite_sf()`, `geo_lite_struct_sf()`, `reverse_geo_lite_sf()`

Examples

```
# bounding box of Germany
bbox_GER <- c(5.86631529, 47.27011137, 15.04193189, 55.09916098)

bbox_GER_sf <- bbox_to_poly(bbox_GER)

library(ggplot2)

ggplot(bbox_GER_sf) +
  geom_sf()

# Extract the bounding box of a sf object
sfobj <- geo_lite_sf("seychelles", points_only = FALSE)

sfobj

# Need at least one non-empty object
if (any(!sf::st_is_empty(sfobj))) {
  bbox <- sf::st_bbox(sfobj)

  bbox

  bbox_sfobj <- bbox_to_poly(bbox)

  ggplot(bbox_sfobj) +
    geom_sf(fill = "lightblue", alpha = 0.5) +
    geom_sf(data = sfobj, fill = "wheat")
}
```

geo_address_lookup *Address lookup API*

Description

The lookup API queries the address and other details of one or multiple OSM objects (node, way, relation) and returns the `tibble` associated with the query; see `geo_address_lookup_sf()` for retrieving the data as a spatial object (`sf` format).

Usage

```
geo_address_lookup(
  osm_ids,
  type = c("N", "W", "R"),
  lat = "lat",
  long = "lon",
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  nominatim_server = "https://nominatim.openstreetmap.org/",
  custom_query = list()
)
```

Arguments

<code>osm_ids</code>	Vector of OSM identifiers as numeric (<code>c(00000, 11111, 22222)</code>).
<code>type</code>	Character vector of the OSM object type associated with each <code>osm_ids</code> . Possible values are node ("N"), way ("W"), or relation ("R"). If a single value is provided it will be recycled.
<code>lat</code>	Latitude column name in the output data (default "lat").
<code>long</code>	Longitude column name in the output data (default "long").
<code>full_results</code>	Returns all available data from the API service. If FALSE (default) only latitude, longitude and address columns are returned. See also <code>return_addresses</code> .
<code>return_addresses</code>	Return input addresses with results if TRUE.
<code>verbose</code>	If TRUE then detailed logs are output to the console.
<code>nominatim_server</code>	The URL of the Nominatim server to use. Defaults to "https://nominatim.openstreetmap.org/".
<code>custom_query</code>	A named list with API-specific parameters to be used (i.e. <code>list(countrycodes = "US")</code>). See Details .

Details

See <https://nominatim.org/release-docs/latest/api/Lookup/> for additional parameters to be passed to `custom_query`.

Value

A [tibble](#) with the results found by the query.

See Also

[geo_address_lookup_sf\(\)](#).

Address Lookup API: [geo_address_lookup_sf\(\)](#)

Geocoding: [geo_address_lookup_sf\(\)](#), [geo_amenity\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite\(\)](#), [geo_lite_sf\(\)](#), [geo_lite_struct\(\)](#), [geo_lite_struct_sf\(\)](#)

Examples

```
ids <- geo_address_lookup(osm_ids = c(46240148, 34633854), type = "W")

ids

several <- geo_address_lookup(c(146656, 240109189), type = c("R", "N"))
several
```

`geo_address_lookup_sf` *Address lookup API in R*[href=https://CRAN.R-project.org/package=sfsf](https://CRAN.R-project.org/package=sfsf)
format

Description

The lookup API queries the address and other details of one or multiple OSM objects (node, way, relation) and returns the spatial object associated with the query using [sf](#); see [geo_address_lookup\(\)](#) for retrieving the data in [tibble](#) format.

Usage

```
geo_address_lookup_sf(  
  osm_ids,  
  type = c("N", "W", "R"),  
  full_results = FALSE,  
  return_addresses = TRUE,  
  verbose = FALSE,  
  nominatim_server = "https://nominatim.openstreetmap.org/",  
  custom_query = list(),  
  points_only = TRUE  
)
```

Arguments

osm_ids	Vector of OSM identifiers as numeric (c(00000, 11111, 22222)).
type	Character vector of the OSM object type associated with each osm_ids. Possible values are node ("N"), way ("W"), or relation ("R"). If a single value is provided it will be recycled.
full_results	Returns all available data from the API service. If FALSE (default) only address columns are returned. See also return_addresses.
return_addresses	Return input addresses with results if TRUE.
verbose	If TRUE then detailed logs are output to the console.
nominatim_server	The URL of the Nominatim server to use. Defaults to "https://nominatim.openstreetmap.org/".
custom_query	A named list with API-specific parameters to be used (i.e. list(countrycodes = "US")). See Details .
points_only	Logical TRUE/FALSE. Whether to return only spatial points (TRUE, which is the default) or potentially other shapes as provided by the Nominatim API (FALSE). See About Geometry Types .

Details

See <https://nominatim.org/release-docs/latest/api/Lookup/> for additional parameters to be passed to custom_query.

Value

A `sf` object with the results.

About Geometry Types

The parameter `points_only` specifies whether the function results will be points (all Nominatim results are guaranteed to have at least point geometry) or possibly other spatial objects.

Note that the type of geometry returned in case of `points_only = FALSE` will depend on the object being geocoded:

- Administrative areas, major buildings and the like will be returned as polygons.
- Rivers, roads and their like as lines.
- Amenities may be points even in case of a `points_only = FALSE` call.

The function is vectorized, allowing for multiple addresses to be geocoded; in case of `points_only = FALSE` multiple geometry types may be returned.

See Also

[geo_address_lookup\(\)](#).

Address Lookup API: [geo_address_lookup\(\)](#)

Geocoding: [geo_address_lookup\(\)](#), [geo_amenity\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite\(\)](#), [geo_lite_sf\(\)](#), [geo_lite_struct\(\)](#), [geo_lite_struct_sf\(\)](#)

Get `sf` objects: [bbox_to_poly\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite_sf\(\)](#), [geo_lite_struct_sf\(\)](#), [reverse_geo_lite_sf\(\)](#)

Examples

```
# Notre Dame Cathedral, Paris

NotreDame <- geo_address_lookup_sf(osm_ids = 201611261, type = "W")

# Need at least one non-empty object
if (any(!sf::st_is_empty(NotreDame))) {
  library(ggplot2)

  ggplot(NotreDame) +
    geom_sf()
}

NotreDame_poly <- geo_address_lookup_sf(201611261,
  type = "W",
  points_only = FALSE
)

if (any(!sf::st_is_empty(NotreDame_poly))) {
  ggplot(NotreDame_poly) +
    geom_sf()
}

# It is vectorized

several <- geo_address_lookup_sf(c(146656, 240109189), type = c("R", "N"))
several
```

geo_amenity

Geocode amenities

Description

Searches [amenities](#) as defined by OpenStreetMap in a restricted area defined by a bounding box in the form (`<xmin>`, `<ymin>`, `<xmax>`, `<ymax>`) and returns the [tibble](#) associated with the query; see [geo_amenity_sf\(\)](#) for retrieving the data as a spatial object (`sf` format).

Usage

```

geo_amenity(
  bbox,
  amenity,
  lat = "lat",
  long = "lon",
  limit = 1,
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  nominatim_server = "https://nominatim.openstreetmap.org/",
  progressbar = TRUE,
  custom_query = list(),
  strict = FALSE
)

```

Arguments

<code>bbox</code>	The bounding box (viewbox) used to limit the search. It could be: <ul style="list-style-type: none"> • A numeric vector of longitude (x) and latitude (y) (xmin, ymin, xmax, ymax). See Details. • A <code>sf</code> or <code>sfc</code> object.
<code>amenity</code>	A character (or a vector of characters) with the amenities to be geolocated (i.e. <code>c("pub", "restaurant")</code>). See osm_amenities .
<code>lat</code>	Latitude column name in the output data (default "lat").
<code>long</code>	Longitude column name in the output data (default "long").
<code>limit</code>	Maximum number of results to return per input address. Note that each query returns a maximum of 50 results.
<code>full_results</code>	Returns all available data from the API service. If FALSE (default) only latitude, longitude and address columns are returned. See also <code>return_addresses</code> .
<code>return_addresses</code>	Return input addresses with results if TRUE.
<code>verbose</code>	If TRUE then detailed logs are output to the console.
<code>nominatim_server</code>	The URL of the Nominatim server to use. Defaults to "https://nominatim.openstreetmap.org/".
<code>progressbar</code>	Logical. If TRUE displays a progress bar to indicate the progress of the function.
<code>custom_query</code>	A named list with API-specific parameters to be used (i.e. <code>list(countrycodes = "US")</code>). See Details .
<code>strict</code>	Logical TRUE/FALSE. Force the results to be included inside the <code>bbox</code> . Note that Nominatim default behavior may return results located outside the provided bounding box.

Details

Bounding boxes can be located using online tools such as <https://boundingbox.klokantech.com/>.

For a full list of valid amenities see <https://wiki.openstreetmap.org/wiki/Key:amenity> and [osm_amenities](#).

See <https://nominatim.org/release-docs/latest/api/Search/> for additional parameters to be passed to `custom_query`.

Value

A [tibble](#) with the results found by the query.

See Also

Other amenity: [geo_amenity_sf\(\)](#), [osm_amenities](#)

Geocoding: [geo_address_lookup\(\)](#), [geo_address_lookup_sf\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite\(\)](#), [geo_lite_sf\(\)](#), [geo_lite_struct\(\)](#), [geo_lite_struct_sf\(\)](#)

Examples

```
# Times Square, NY, USA
bbox <- c(
  -73.9894467311, 40.75573629,
  -73.9830630737, 40.75789245
)

geo_amenity(
  bbox = bbox,
  amenity = "restaurant"
)

# Several amenities
geo_amenity(
  bbox = bbox,
  amenity = c("restaurant", "pub")
)

# Increase limit and use with strict
geo_amenity(
  bbox = bbox,
  amenity = c("restaurant", "pub"),
  limit = 10,
  strict = TRUE
)
```

geo_amenity_sf	<i>Geocode amenities in R</i> https://CRAN.R-project.org/package=sfsf <i>format</i>
----------------	--

Description

Searches [amenities](#) as defined by OpenStreetMap in a restricted area defined by a bounding box in the form (<xmin>, <ymin>, <xmax>, <ymax>) and returns the spatial object associated with the query using [sf](#); see [geo_amenity\(\)](#) for retrieving the data in [tibble](#) format.

Usage

```
geo_amenity_sf(
  bbox,
  amenity,
  limit = 1,
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  nominatim_server = "https://nominatim.openstreetmap.org/",
  progressbar = TRUE,
  custom_query = list(),
  strict = FALSE,
  points_only = TRUE
)
```

Arguments

bbox	The bounding box (viewbox) used to limit the search. It could be: <ul style="list-style-type: none"> • A numeric vector of longitude (x) and latitude (y) (xmin, ymin, xmax, ymax). See Details. • A sf or sfc object.
amenity	A character (or a vector of characters) with the amenities to be geolocated (i.e. c("pub", "restaurant")). See osm_amenities .
limit	Maximum number of results to return per input address. Note that each query returns a maximum of 50 results.
full_results	Returns all available data from the API service. If FALSE (default) only latitude, longitude and address columns are returned. See also return_addresses.
return_addresses	Return input addresses with results if TRUE.
verbose	If TRUE then detailed logs are output to the console.
nominatim_server	The URL of the Nominatim server to use. Defaults to "https://nominatim.openstreetmap.org/".
progressbar	Logical. If TRUE displays a progress bar to indicate the progress of the function.

custom_query	A named list with API-specific parameters to be used (i.e. <code>list(countrycodes = "US")</code>). See Details .
strict	Logical TRUE/FALSE. Force the results to be included inside the bbox. Note that Nominatim default behavior may return results located outside the provided bounding box.
points_only	Logical TRUE/FALSE. Whether to return only spatial points (TRUE, which is the default) or potentially other shapes as provided by the Nominatim API (FALSE). See About Geometry Types .

Details

Bounding boxes can be located using online tools such as <https://boundingbox.klokantech.com/>.

For a full list of valid amenities see <https://wiki.openstreetmap.org/wiki/Key:amenity> and [osm_amenities](#).

See <https://nominatim.org/release-docs/latest/api/Search/> for additional parameters to be passed to custom_query.

Value

A `sf` object with the results.

About Geometry Types

The parameter `points_only` specifies whether the function results will be points (all Nominatim results are guaranteed to have at least point geometry) or possibly other spatial objects.

Note that the type of geometry returned in case of `points_only = FALSE` will depend on the object being geocoded:

- Administrative areas, major buildings and the like will be returned as polygons.
- Rivers, roads and their like as lines.
- Amenities may be points even in case of a `points_only = FALSE` call.

The function is vectorized, allowing for multiple addresses to be geocoded; in case of `points_only = FALSE` multiple geometry types may be returned.

See Also

Other amenity: [geo_amenity\(\)](#), [osm_amenities](#)

Geocoding: [geo_address_lookup\(\)](#), [geo_address_lookup_sf\(\)](#), [geo_amenity\(\)](#), [geo_lite\(\)](#), [geo_lite_sf\(\)](#), [geo_lite_struct\(\)](#), [geo_lite_struct_sf\(\)](#)

Get `sf` objects: [bbox_to_poly\(\)](#), [geo_address_lookup_sf\(\)](#), [geo_lite_sf\(\)](#), [geo_lite_struct_sf\(\)](#), [reverse_geo_lite_sf\(\)](#)

Examples

```
# Usera, Madrid

library(ggplot2)
mad <- geo_lite_sf("Usera, Madrid, Spain", points_only = FALSE)

# Restaurants, pubs and schools

rest_pub <- geo_amenity_sf(mad, c("restaurant", "pub", "school"),
  limit = 50
)

if (any(!sf::st_is_empty(rest_pub))) {
  ggplot(mad) +
    geom_sf() +
    geom_sf(data = rest_pub, aes(color = query, shape = query))
}
```

geo_lite

Address search API (free-form query)

Description

Geocodes addresses given as character values and returns the [tibble](#) associated with the query; see [geo_lite_sf\(\)](#) for retrieving the data as a spatial object ([sf](#) format).

Corresponds to the **free-form query** search described in the [API endpoint](#).

Usage

```
geo_lite(
  address,
  lat = "lat",
  long = "lon",
  limit = 1,
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  nominatim_server = "https://nominatim.openstreetmap.org/",
  progressbar = TRUE,
  custom_query = list()
)
```

Arguments

address	character with single line address, e.g. ("1600 Pennsylvania Ave NW, Washington") or a vector of addresses (c("Madrid", "Barcelona")).
lat	Latitude column name in the output data (default "lat").
long	Longitude column name in the output data (default "long").
limit	Maximum number of results to return per input address. Note that each query returns a maximum of 50 results.
full_results	Returns all available data from the API service. If FALSE (default) only latitude, longitude and address columns are returned. See also return_addresses.
return_addresses	Return input addresses with results if TRUE.
verbose	If TRUE then detailed logs are output to the console.
nominatim_server	The URL of the Nominatim server to use. Defaults to "https://nominatim.openstreetmap.org/".
progressbar	Logical. If TRUE displays a progress bar to indicate the progress of the function.
custom_query	A named list with API-specific parameters to be used (i.e. list(countrycodes = "US")). See Details .

Details

See <https://nominatim.org/release-docs/latest/api/Search/> for additional parameters to be passed to custom_query.

Value

A [tibble](#) with the results found by the query.

See Also

[geo_lite_sf\(\)](#), [tidygeocoder::geo\(\)](#).

Geocoding: [geo_address_lookup\(\)](#), [geo_address_lookup_sf\(\)](#), [geo_amenity\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite_sf\(\)](#), [geo_lite_struct\(\)](#), [geo_lite_struct_sf\(\)](#)

Examples

```
geo_lite("Madrid, Spain")

# Several addresses
geo_lite(c("Madrid", "Barcelona"))

# With options: restrict search to USA
geo_lite(c("Madrid", "Barcelona"),
  custom_query = list(countrycodes = "US"),
  full_results = TRUE
)
```

geo_lite_sf	<i>Address search API in R</i> https://CRAN.R-project.org/package=sf <i>format (free-form query)</i>
-------------	---

Description

Geocodes addresses and returns the corresponding spatial object. The query output is provided in **sf** format; see `geo_lite()` for retrieving the data in **tibble** format.

Corresponds to the **free-form query** search described in the **API endpoint**.

Usage

```
geo_lite_sf(
  address,
  limit = 1,
  return_addresses = TRUE,
  full_results = FALSE,
  verbose = FALSE,
  progressbar = TRUE,
  nominatim_server = "https://nominatim.openstreetmap.org/",
  custom_query = list(),
  points_only = TRUE
)
```

Arguments

address	character with single line address, e.g. ("1600 Pennsylvania Ave NW, Washington") or a vector of addresses (c("Madrid", "Barcelona")).
limit	Maximum number of results to return per input address. Note that each query returns a maximum of 50 results.
return_addresses	Return input addresses with results if TRUE.
full_results	Returns all available data from the API service. If FALSE (default) only address columns are returned. See also return_addresses.
verbose	If TRUE then detailed logs are output to the console.
progressbar	Logical. If TRUE displays a progress bar to indicate the progress of the function.
nominatim_server	The URL of the Nominatim server to use. Defaults to "https://nominatim.openstreetmap.org/".
custom_query	A named list with API-specific parameters to be used (i.e. list(countrycodes = "US")). See Details .
points_only	Logical TRUE/FALSE. Whether to return only spatial points (TRUE, which is the default) or potentially other shapes as provided by the Nominatim API (FALSE). See About Geometry Types .

Details

See <https://nominatim.org/release-docs/latest/api/Search/> for additional parameters to be passed to `custom_query`.

Value

A `sf` object with the results.

About Geometry Types

The parameter `points_only` specifies whether the function results will be points (all Nominatim results are guaranteed to have at least point geometry) or possibly other spatial objects.

Note that the type of geometry returned in case of `points_only = FALSE` will depend on the object being geocoded:

- Administrative areas, major buildings and the like will be returned as polygons.
- Rivers, roads and their like as lines.
- Amenities may be points even in case of a `points_only = FALSE` call.

The function is vectorized, allowing for multiple addresses to be geocoded; in case of `points_only = FALSE` multiple geometry types may be returned.

See Also

[geo_lite\(\)](#).

Geocoding: [geo_address_lookup\(\)](#), [geo_address_lookup_sf\(\)](#), [geo_amenity\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite\(\)](#), [geo_lite_struct\(\)](#), [geo_lite_struct_sf\(\)](#)

Get `sf` objects: [bbox_to_poly\(\)](#), [geo_address_lookup_sf\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite_struct_sf\(\)](#), [reverse_geo_lite_sf\(\)](#)

Examples

```
# Map - Points
library(ggplot2)

string <- "Statue of Liberty, NY, USA"
sol <- geo_lite_sf(string)

if (any(!sf::st_is_empty(sol))) {
  ggplot(sol) +
    geom_sf()
}

sol_poly <- geo_lite_sf(string, points_only = FALSE)

if (any(!sf::st_is_empty(sol_poly))) {
  ggplot(sol_poly) +
    geom_sf() +
```

```
    geom_sf(data = sol, color = "red")
  }
# Several results

madrid <- geo_lite_sf("Comunidad de Madrid, Spain",
  limit = 2,
  points_only = FALSE, full_results = TRUE
)

if (any(!sf::st_is_empty(madrid))) {
  ggplot(madrid) +
    geom_sf(fill = NA)
}
```

geo_lite_struct

Address search API (structured query)

Description

Geocodes addresses already split into components and returns the [tibble](#) associated with the query; see [geo_lite_struct_sf\(\)](#) for retrieving the data as a spatial object ([sf](#) format).

Corresponds to the **structured query** search described in the [API endpoint](#). For performing a free-form search use [geo_lite\(\)](#).

Usage

```
geo_lite_struct(
  amenity = NULL,
  street = NULL,
  city = NULL,
  county = NULL,
  state = NULL,
  country = NULL,
  postalcode = NULL,
  lat = "lat",
  long = "lon",
  limit = 1,
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  nominatim_server = "https://nominatim.openstreetmap.org/",
  custom_query = list()
)
```

Arguments

amenity	Name and/or type of POI, see also geo_amenity .
street	House number and street name.
city	City.
county	County.
state	State.
country	Country.
postalcode	Postal Code.
lat	Latitude column name in the output data (default "lat").
long	Longitude column name in the output data (default "long").
limit	Maximum number of results to return per input address. Note that each query returns a maximum of 50 results.
full_results	Returns all available data from the API service. If FALSE (default) only latitude, longitude and address columns are returned. See also <code>return_addresses</code> .
return_addresses	Return input addresses with results if TRUE.
verbose	If TRUE then detailed logs are output to the console.
nominatim_server	The URL of the Nominatim server to use. Defaults to "https://nominatim.openstreetmap.org/".
custom_query	A named list with API-specific parameters to be used (i.e. <code>list(countrycodes = "US")</code>). See Details .

Details

The structured form of the search query allows you to look up an address that is already split into its components. Each parameter represents a field of the address. All parameters are optional. You should only use the ones that are relevant for the address you want to geocode.

See <https://nominatim.org/release-docs/latest/api/Search/> for additional parameters to be passed to `custom_query`.

Value

A [tibble](#) with the results found by the query.

See Also

[geo_lite_struct_sf\(\)](#), [tidygeocoder::geo\(\)](#).

Geocoding: [geo_address_lookup\(\)](#), [geo_address_lookup_sf\(\)](#), [geo_amenity\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite\(\)](#), [geo_lite_sf\(\)](#), [geo_lite_struct_sf\(\)](#)

Examples

```
pl_mayor <- geo_lite_struct(
  street = "Plaza Mayor", country = "Spain",
  limit = 50, full_results = TRUE
)

dplyr::glimpse(pl_mayor)
```

geo_lite_struct_sf *Address search API in R*[href=https://CRAN.R-project.org/package=sfsf](https://CRAN.R-project.org/package=sfsf)
format (structured query)

Description

Geocodes addresses already split into components and returns the corresponding spatial object. The query output is provided in **sf** format; see `geo_lite_struct()` for retrieving the data in **tibble** format.

Corresponds to the **structured query** search described in the **API endpoint**. For performing a free-form search use `geo_lite_sf()`.

Usage

```
geo_lite_struct_sf(
  amenity = NULL,
  street = NULL,
  city = NULL,
  county = NULL,
  state = NULL,
  country = NULL,
  postalcode = NULL,
  limit = 1,
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  nominatim_server = "https://nominatim.openstreetmap.org/",
  custom_query = list(),
  points_only = TRUE
)
```

Arguments

amenity	Name and/or type of POI, see also geo_amenity .
street	House number and street name.

city	City.
county	County.
state	State.
country	Country.
postalcode	Postal Code.
limit	Maximum number of results to return per input address. Note that each query returns a maximum of 50 results.
full_results	Returns all available data from the API service. If FALSE (default) only latitude, longitude and address columns are returned. See also return_addresses.
return_addresses	Return input addresses with results if TRUE.
verbose	If TRUE then detailed logs are output to the console.
nominatim_server	The URL of the Nominatim server to use. Defaults to "https://nominatim.openstreetmap.org/".
custom_query	A named list with API-specific parameters to be used (i.e. list(countrycodes = "US")). See Details .
points_only	Logical TRUE/FALSE. Whether to return only spatial points (TRUE, which is the default) or potentially other shapes as provided by the Nominatim API (FALSE). See About Geometry Types .

Details

The structured form of the search query allows you to look up an address that is already split into its components. Each parameter represents a field of the address. All parameters are optional. You should only use the ones that are relevant for the address you want to geocode.

See <https://nominatim.org/release-docs/latest/api/Search/> for additional parameters to be passed to custom_query.

Value

A `sf` object with the results.

About Geometry Types

The parameter `points_only` specifies whether the function results will be points (all Nominatim results are guaranteed to have at least point geometry) or possibly other spatial objects.

Note that the type of geometry returned in case of `points_only = FALSE` will depend on the object being geocoded:

- Administrative areas, major buildings and the like will be returned as polygons.
- Rivers, roads and their like as lines.
- Amenities may be points even in case of a `points_only = FALSE` call.

The function is vectorized, allowing for multiple addresses to be geocoded; in case of `points_only = FALSE` multiple geometry types may be returned.

See Also

[geo_lite_struct\(\)](#).

Geocoding: [geo_address_lookup\(\)](#), [geo_address_lookup_sf\(\)](#), [geo_amenity\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite\(\)](#), [geo_lite_sf\(\)](#), [geo_lite_struct\(\)](#)

Get `sf` objects: [bbox_to_poly\(\)](#), [geo_address_lookup_sf\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite_sf\(\)](#), [reverse_geo_lite_sf\(\)](#)

Examples

```
# Map

pl_mayor <- geo_lite_struct_sf(
  street = "Plaza Mayor",
  county = "Comunidad de Madrid",
  country = "Spain", limit = 50,
  full_results = TRUE, verbose = TRUE
)

# Outline
ccaa <- geo_lite_sf("Comunidad de Madrid, Spain", points_only = FALSE)

library(ggplot2)

if (any(!sf::st_is_empty(pl_mayor), !sf::st_is_empty(ccaa))) {
  ggplot(ccaa) +
    geom_sf() +
    geom_sf(data = pl_mayor, aes(shape = addresstype, color = addresstype))
}
```

osm_amenities

OpenStreetMap amenity database

Description

Database with the list of amenities available on OpenStreetMap.

Format

A [tibble](#) with 136 rows and fields:

category The category of the amenity.

amenity The value of the amenity.

comment A brief description of the type of amenity.

Note

Data extracted on **03 April 2024**.

Source

<https://wiki.openstreetmap.org/wiki/Key:amenity>

See Also

Other amenity: [geo_amenity\(\)](#), [geo_amenity_sf\(\)](#)

Examples

```
data("osm_amenities")  
  
osm_amenities
```

reverse_geo_lite	<i>Reverse geocoding API</i>
------------------	------------------------------

Description

Generates an address from latitude and longitude (latitudes in $[-90, 90]$ and longitudes in $[-180, 180]$), and returns the [tibble](#) associated with the query; see [reverse_geo_lite_sf\(\)](#) for retrieving the data as a spatial object ([sf](#) format).

Usage

```
reverse_geo_lite(  
  lat,  
  long,  
  address = "address",  
  full_results = FALSE,  
  return_coords = TRUE,  
  verbose = FALSE,  
  nominatim_server = "https://nominatim.openstreetmap.org/",  
  progressbar = TRUE,  
  custom_query = list()  
)
```

Arguments

lat	Latitude values in numeric format. Must be in the range $[-90, 90]$.
long	Longitude values in numeric format. Must be in the range $[-180, 180]$.
address	Address column name in the output data (default "address").
full_results	Returns all available data from the API service. If FALSE (default) only latitude, longitude and address columns are returned. See also return_addresses .

return_coords	Return input coordinates with results if TRUE.
verbose	If TRUE then detailed logs are output to the console.
nominatim_server	The URL of the Nominatim server to use. Defaults to "https://nominatim.openstreetmap.org/".
progressbar	Logical. If TRUE displays a progress bar to indicate the progress of the function.
custom_query	API-specific parameters to be used, passed as a named list (ie. list(zoom = 3)). See Details .

Details

See <https://nominatim.org/release-docs/latest/api/Reverse/> for additional parameters to be passed to custom_query.

Value

A [tibble](#) with the results found by the query.

About Zooming

Use the option custom_query = list(zoom = 3) to adjust the output. Some equivalences on terms of zoom:

zoom	address_detail
3	country
5	state
8	county
10	city
14	suburb
16	major streets
17	major and minor streets
18	building

See Also

[reverse_geo_lite_sf\(\)](#), [tidygeocoder::reverse_geo\(\)](#).

Reverse geocoding coordinates: [reverse_geo_lite_sf\(\)](#)

Examples

```
reverse_geo_lite(lat = 40.75728, long = -73.98586)

# Several coordinates
reverse_geo_lite(lat = c(40.75728, 55.95335), long = c(-73.98586, -3.188375))

# With options: zoom to country level
sev <- reverse_geo_lite(
```

```

lat = c(40.75728, 55.95335), long = c(-73.98586, -3.188375),
custom_query = list(zoom = 0, extratags = TRUE),
verbose = TRUE, full_results = TRUE
)

dplyr::glimpse(sev)

```

reverse_geo_lite_sf *Reverse geocoding API in R* [hrefhttps://CRAN.R-project.org/package=sf](https://CRAN.R-project.org/package=sf) *sf format*

Description

Generates an address from latitude and longitude (latitudes in $[-90, 90]$ and longitudes in $[-180, 180]$), and returns the spatial object associated with the query using **sf**; see `reverse_geo_lite()` for retrieving the data in **tibble** format.

Usage

```

reverse_geo_lite_sf(
  lat,
  long,
  address = "address",
  full_results = FALSE,
  return_coords = TRUE,
  verbose = FALSE,
  nominatim_server = "https://nominatim.openstreetmap.org/",
  progressbar = TRUE,
  custom_query = list(),
  points_only = TRUE
)

```

Arguments

lat	Latitude values in numeric format. Must be in the range $[-90, 90]$.
long	Longitude values in numeric format. Must be in the range $[-180, 180]$.
address	Address column name in the output data (default "address").
full_results	Returns all available data from the API service. If FALSE (default) only latitude, longitude and address columns are returned. See also <code>return_addresses</code> .
return_coords	Return input coordinates with results if TRUE.
verbose	If TRUE then detailed logs are output to the console.
nominatim_server	The URL of the Nominatim server to use. Defaults to "https://nominatim.openstreetmap.org/".
progressbar	Logical. If TRUE displays a progress bar to indicate the progress of the function.

custom_query	API-specific parameters to be used, passed as a named list (ie. <code>list(zoom = 3)</code>). See Details .
points_only	Logical TRUE/FALSE. Whether to return only spatial points (TRUE, which is the default) or potentially other shapes as provided by the Nominatim API (FALSE). See About Geometry Types .

Details

See <https://nominatim.org/release-docs/latest/api/Reverse/> for additional parameters to be passed to custom_query.

Value

A `sf` object with the results.

About Zooming

Use the option `custom_query = list(zoom = 3)` to adjust the output. Some equivalences on terms of zoom:

zoom	address_detail
3	country
5	state
8	county
10	city
14	suburb
16	major streets
17	major and minor streets
18	building

About Geometry Types

The parameter `points_only` specifies whether the function results will be points (all Nominatim results are guaranteed to have at least point geometry) or possibly other spatial objects.

Note that the type of geometry returned in case of `points_only = FALSE` will depend on the object being geocoded:

- Administrative areas, major buildings and the like will be returned as polygons.
- Rivers, roads and their like as lines.
- Amenities may be points even in case of a `points_only = FALSE` call.

The function is vectorized, allowing for multiple addresses to be geocoded; in case of `points_only = FALSE` multiple geometry types may be returned.

See Also

[reverse_geo_lite\(\)](#).

Reverse geocoding coordinates: [reverse_geo_lite\(\)](#)

Get `sf` objects: [bbox_to_poly\(\)](#), [geo_address_lookup_sf\(\)](#), [geo_amenity_sf\(\)](#), [geo_lite_sf\(\)](#), [geo_lite_struct_sf\(\)](#)

Examples

```
library(ggplot2)

# Coliseum coords
col_lon <- 12.49309
col_lat <- 41.89026

# Coliseum as polygon
col_sf <- reverse_geo_lite_sf(
  lat = col_lat,
  lon = col_lon,
  points_only = FALSE
)

dplyr::glimpse(col_sf)

if (any(!sf::st_is_empty(col_sf))) {
  ggplot(col_sf) +
    geom_sf()
}

# City of Rome - same coords with zoom 10

rome_sf <- reverse_geo_lite_sf(
  lat = col_lat,
  lon = col_lon,
  custom_query = list(zoom = 10),
  points_only = FALSE
)

dplyr::glimpse(rome_sf)

if (any(!sf::st_is_empty(rome_sf))) {
  ggplot(rome_sf) +
    geom_sf()
}
```

Index

- * **amenity**
 - geo_amenity, 7
 - geo_amenity_sf, 10
 - osm_amenities, 20
 - * **datasets**
 - osm_amenities, 20
 - * **geocoding**
 - geo_address_lookup, 4
 - geo_address_lookup_sf, 5
 - geo_amenity, 7
 - geo_amenity_sf, 10
 - geo_lite, 12
 - geo_lite_sf, 14
 - geo_lite_struct, 16
 - geo_lite_struct_sf, 18
 - * **lookup**
 - geo_address_lookup, 4
 - geo_address_lookup_sf, 5
 - * **reverse**
 - reverse_geo_lite, 21
 - reverse_geo_lite_sf, 23
 - * **spatial**
 - bbox_to_poly, 2
 - geo_address_lookup_sf, 5
 - geo_amenity_sf, 10
 - geo_lite_sf, 14
 - geo_lite_struct_sf, 18
 - reverse_geo_lite_sf, 23
- amenities, 7, 10
- bbox_to_poly, 2, 7, 11, 15, 20, 25
- geo_address_lookup, 4, 6, 7, 9, 11, 13, 15, 17, 20
- geo_address_lookup(), 5, 6
- geo_address_lookup_sf, 3, 5, 5, 9, 11, 13, 15, 17, 20, 25
- geo_address_lookup_sf(), 4, 5
- geo_amenity, 5, 7, 7, 11, 13, 15, 17, 18, 20, 21
- geo_amenity(), 10
- geo_amenity_sf, 3, 5, 7, 9, 10, 13, 15, 17, 20, 21, 25
- geo_amenity_sf(), 7
- geo_lite, 5, 7, 9, 11, 12, 15, 17, 20
- geo_lite(), 14–16
- geo_lite_sf, 3, 5, 7, 9, 11, 13, 14, 17, 20, 25
- geo_lite_sf(), 12, 13, 18
- geo_lite_struct, 5, 7, 9, 11, 13, 15, 16, 20
- geo_lite_struct(), 18, 20
- geo_lite_struct_sf, 3, 5, 7, 9, 11, 13, 15, 17, 18, 25
- geo_lite_struct_sf(), 16, 17
- osm_amenities, 8–11, 20
- reverse_geo_lite, 21, 25
- reverse_geo_lite(), 23, 25
- reverse_geo_lite_sf, 3, 7, 11, 15, 20, 22, 23
- reverse_geo_lite_sf(), 21, 22
- sf, 3, 4, 6–8, 10–12, 15, 16, 19–21, 24, 25
- sf::st_as_sfc(), 3
- sf::st_sfc(), 3
- sfc, 2, 3, 8, 10
- st_crs, 2
- tibble, 4, 5, 7, 9, 10, 12–14, 16–18, 20–23
- tidygeocoder::geo(), 13, 17
- tidygeocoder::reverse_geo(), 22