Package ‘nominatimlite’

June 10, 2022

Type Package

Title Interface with 'Nominatim' API Service

Version 0.1.6

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, search for amenities and return spatial objects on 'sf' format.

License MIT + file LICENSE


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

Depends R (>= 3.6.0)

Imports dplyr (>= 1.0.0), jsonlite (>= 1.7.0), rlang (>= 0.4.9), sf (>= 0.9.0), tibble (>= 3.0.3), utils

Suggests ggplot2 (>= 3.0.0), knitr, osmdata, rmarkdown, testthat (>= 3.0.0), tidygeocoder

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

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

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

X-schema.org-applicationCategory cartography

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

NeedsCompilation no
Author  Diego Hernangómez [aut, cre, cph]
          (<https://orcid.org/0000-0001-8457-4658>),
          Jindra Lacko [ctb, rev]

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

Repository  CRAN

Date/Publication  2022-06-10 07:40:02 UTC

R topics documented:

bbox_to_poly  Create a bounding box sf object

Description
Create a sf polygon 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 c(xmin, ymin, xmax, ymax)

xmin, ymin, xmax, ymax
alternatively, you can use these named parameters instead of bbox

crs                  coordinate reference system, something suitable as input to st_crs

Details
Bounding boxes can be located using different online tools, as Bounding Box Tool.
**Value**

A sf object

**See Also**

`sf::st_as_sfc()`

Other spatial: `geo_address_lookup_sf()`, `geo_amenity_sf()`, `geo_lite_sf()`, `reverse_geo_lite_sf()`

Other amenity: `geo_amenity_sf()`, `geo_amenity()`, `osm_amenities`

**Examples**

```r
# 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
Texas <- geo_lite_sf("Texas", points_only = FALSE)
bbox <- sf::st_bbox(Texas)

bbox

bbox_Texas <- bbox_to_poly(bbox)

ggplot(bbox_Texas) +
  geom_sf(col = "red") +
  geom_sf(data = Texas)
```

---

**Description**

Geocodes addresses for OSM objects, identified with the OSM Id.
Usage

```r
geo_address_lookup(
  osm_ids,
  type = c("N", "W", "R"),
  lat = "lat",
  long = "lon",
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  custom_query = list()
)
```

Arguments

- `osm_ids` vector of OSM identifiers (`c(00000, 11111, 22222)`).
- `type` vector of the type of the OSM type associated to each `osm_ids`. Possible values are node ("N"), way ("W") or relation ("R"). If a single value is provided it would be recycled.
- `lat` latitude column name (i.e. "lat").
- `long` longitude column name (i.e. "long").
- `full_results` returns all available data from the geocoding service if TRUE. If FALSE (default) then only latitude and longitude columns are returned from the geocoding service.
- `return_addresses` return input addresses with results if TRUE. Note that most services return the input addresses with `full_results = TRUE` and setting `return_addresses` to FALSE does not prevent this.
- `verbose` if TRUE then detailed logs are output to the console. FALSE is default. Can be set permanently with `options(tidygeocoder.verbose = TRUE)`
- `custom_query` API-specific parameters to be used, passed as a named list (i.e. `list(countrycodes = "US")`). See Details.

Details

See [https://nominatim.org/release-docs/develop/api/Lookup/](https://nominatim.org/release-docs/develop/api/Lookup/) for additional parameters to be passed to `custom_query`.

Value

A tibble with the results.

See Also

Other geocoding: `geo_amenity()`, `geo_lite()`, `reverse_geo_lite()`
Other lookup: `geo_address_lookup_sf()`
Examples

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

ids
```

---

**geo_address_lookup_sf**  
*Get spatial objects from OSM ids*

---

**Description**

This function allows you to extract the spatial objects for specific OSM objects.

**Usage**

```r
geo_address_lookup_sf(
  osm_ids,  
  type = c("N", "W", "R"),  
  full_results = FALSE,  
  return_addresses = TRUE,  
  verbose = FALSE,  
  custom_query = list(),  
  points_only = TRUE
)
```

**Arguments**

- `osm_ids`: vector of OSM identifiers (c(00000, 11111, 22222)).
- `type`: vector of the type of the OSM type associated to each `osm_ids`. Possible values are node ("N"), way ("W") or relation ("R"). If a single value is provided it would be recycled.
- `full_results`: returns all available data from the geocoding service if TRUE. If FALSE (default) then only latitude and longitude columns are returned from the geocoding service.
- `return_addresses`: return input addresses with results if TRUE. Note that most services return the input addresses with `full_results = TRUE` and setting `return_addresses` to FALSE does not prevent this.
- `verbose`: if TRUE then detailed logs are output to the console. FALSE is default. Can be set permanently with `options(tidygeocoder.verbose = TRUE)`
custom_query  API-specific parameters to be used, passed as a named list (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).

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.

See Also

Other spatial: bbox_to_poly(), geo_amenity_sf(), geo_lite_sf(), reverse_geo_lite_sf()

Other lookup: geo_address_lookup()

Examples

# Notre Dame Cathedral, Paris
NotreDame <- geo_address_lookup_sf(
  osm_ids = c(201611261),
  type = c("W")
)
library(ggplot2)
ggplot(NotreDame) +
  geom_sf()

NotreDame_poly <- geo_address_lookup_sf(
  osm_ids = c(201611261),
  type = c("W"),
  points_only = FALSE
)
ggplot(NotreDame_poly) +
  geom_sf()
geo_amenity

**Geocode amenities**

**Description**

This function searches amenities as defined by OpenStreetMap on a restricted area defined by a bounding box in the form of (<min_latitude>, <min_longitude>, <max_latitude>, <max_longitude>).

**Usage**

```r
geo_amenity(
  bbox,
  amenity,
  lat = "lat",
  long = "lon",
  limit = 1,
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  custom_query = list(),
  strict = FALSE
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bbox</code></td>
<td>A numeric vector of latitude and longitude (&lt;min_latitude&gt;, &lt;min_longitude&gt;,</td>
</tr>
<tr>
<td></td>
<td>&lt;max_latitude&gt;, &lt;max_longitude&gt;) that restrict the search area. See Details.</td>
</tr>
<tr>
<td><code>amenity</code></td>
<td>A character of a vector of character with the amenities to be geolocated (i.e.</td>
</tr>
<tr>
<td></td>
<td>c(&quot;pub&quot;, &quot;restaurant&quot;). See Details or osm_amenities.</td>
</tr>
<tr>
<td><code>lat</code></td>
<td>latitude column name (i.e. &quot;lat&quot;).</td>
</tr>
<tr>
<td><code>long</code></td>
<td>longitude column name (i.e. &quot;lon&quot;).</td>
</tr>
<tr>
<td><code>limit</code></td>
<td>maximum number of results to return per input address. Note that each query</td>
</tr>
<tr>
<td></td>
<td>returns a maximum of 50 results.</td>
</tr>
<tr>
<td><code>full_results</code></td>
<td>returns all available data from the geocoding service if TRUE. If FALSE (de-</td>
</tr>
<tr>
<td></td>
<td>fault) then only latitude and longitude columns are returned from the geocoding</td>
</tr>
<tr>
<td></td>
<td>service.</td>
</tr>
<tr>
<td><code>return_addresses</code></td>
<td>return input addresses with results if TRUE. Note that most services return</td>
</tr>
<tr>
<td></td>
<td>the input addresses with full_results = TRUE and setting return_addresses to</td>
</tr>
<tr>
<td></td>
<td>FALSE does not prevent this.</td>
</tr>
<tr>
<td><code>verbose</code></td>
<td>if TRUE then detailed logs are output to the console. FALSE is default. Can</td>
</tr>
<tr>
<td></td>
<td>be set permanently with options(tidygeocoder.verbose = TRUE)</td>
</tr>
<tr>
<td><code>custom_query</code></td>
<td>API-specific parameters to be used. See geo_lite().</td>
</tr>
<tr>
<td><code>strict</code></td>
<td>Logical TRUE/FALSE. Force the results to be included inside the bbox. Note</td>
</tr>
<tr>
<td></td>
<td>that Nominatim default behaviour may return results located outside the pro-</td>
</tr>
<tr>
<td></td>
<td>vided bounding box.</td>
</tr>
</tbody>
</table>
Details

Bounding boxes can be located using different online tools, as Bounding Box Tool.

For a full list of valid amenities see https://wiki.openstreetmap.org/wiki/Key:amenity.

Value

A tibble with the results.

See Also

Other amenity: bbox_to_poly(), geo_amenity_sf(), osm_amenities
Other geocoding: geo_address_lookup(), geo_lite(), reverse_geo_lite()

Examples

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

df <- 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

Get spatial objects of amenities

Description

This function searches amenities as defined by OpenStreetMap on a restricted area defined by a bounding box in the form of (<min_latitude>, <min_longitude>, <max_latitude>, <max_longitude>).

Usage

geo_amenity_sf(
  bbox,  # A numeric vector of latitude and longitude (<min_latitude>, <min_longitude>, <max_latitude>, <max_longitude>) that restrict the search area. See Details.
  amenity,  # A character of a vector of character with the amenities to be geolocated (i.e. c("pub", "restaurant")): See Details or osm_amenities.
  limit = 1,  # maximum number of results to return per input address. Note that each query returns a maximum of 50 results.
  full_results = FALSE,  # returns all available data from the geocoding service if TRUE. If FALSE (default) then only latitude and longitude columns are returned from the geocoding service.
  return_addresses = TRUE,  # return input addresses with results if TRUE. Note that most services return the input addresses with full_results = TRUE and setting return_addresses to FALSE does not prevent this.
  verbose = FALSE,  # if TRUE then detailed logs are output to the console. FALSE is default. Can be set permanently with options(tidygeocoder.verbose = TRUE)
  custom_query = list(),  # API-specific parameters to be used. See geo_lite().
  points_only = TRUE,  # 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).
  strict = FALSE)  # Logical TRUE/FALSE. Force the results to be included inside the bbox. Note that Nominatim default behaviour may return results located outside the provided bounding box.

Arguments

bbox  # A numeric vector of latitude and longitude (<min_latitude>, <min_longitude>, <max_latitude>, <max_longitude>) that restrict the search area. See Details.
amenity  # A character of a vector of character with the amenities to be geolocated (i.e. c("pub", "restaurant")): See Details or 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 geocoding service if TRUE. If FALSE (default) then only latitude and longitude columns are returned from the geocoding service.
return_addresses  # return input addresses with results if TRUE. Note that most services return the input addresses with full_results = TRUE and setting return_addresses to FALSE does not prevent this.
verbose  # if TRUE then detailed logs are output to the console. FALSE is default. Can be set permanently with options(tidygeocoder.verbose = TRUE)
custom_query  # API-specific parameters to be used. See geo_lite().
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).
strict  # Logical TRUE/FALSE. Force the results to be included inside the bbox. Note that Nominatim default behaviour may return results located outside the provided bounding box.
Details

Bounding boxes can be located using different online tools, as Bounding Box Tool.
For a full list of valid amenities see [https://wiki.openstreetmap.org/wiki/Key:amenity](https://wiki.openstreetmap.org/wiki/Key:amenity).

Value

A sf object with the results.

See Also

- Other spatial: `bbox_to_poly()`, `geo_address_lookup_sf()`, `geo_lite_sf()`, `reverse_geo_lite_sf()`
- Other amenity: `bbox_to_poly()`, `geo_amenity()`, `osm_amenities`

Examples

# Madrid, Spain
library(ggplot2)
bbox <- c(
  -3.888954, 40.311977,
  -3.517916, 40.643729
)

# Restaurants and pubs
rest_pub <- geo_amenity_sf(bbox,
  c("restaurant", "pub"),
  limit = 50
)
ggplot(rest_pub) +
  geom_sf()

# Hospital as polygon
hosp <- geo_amenity_sf(bbox,
  "hospital",
  points_only = FALSE
)
ggplot(hosp) +
  geom_sf()
Description

Geocodes addresses given as character values.

Usage

```r
geo_lite(
  address,
  lat = "lat",
  long = "lon",
  limit = 1,
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  custom_query = list()
)
```

Arguments

- **address**: single line address (i.e. "1600 Pennsylvania Ave NW, Washington") or a vector of addresses (c("Madrid", "Barcelona").
- **lat**: latitude column name (i.e. "lat").
- **long**: longitude column name (i.e. "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 geocoding service if TRUE. If FALSE (default) then only latitude and longitude columns are returned from the geocoding service.
- **return_addresses**: return input addresses with results if TRUE. Note that most services return the input addresses with full_results = TRUE and setting return_addresses to FALSE does not prevent this.
- **verbose**: if TRUE then detailed logs are output to the console. FALSE is default. Can be set permanently with options(tidygeocoder.verbose = TRUE)
- **custom_query**: API-specific parameters to be used, passed as a named list (i.e. list(countrycodes = "US")). See Details.

Details

See [https://nominatim.org/release-docs/latest/api/Search/](https://nominatim.org/release-docs/latest/api/Search/) for additional parameters to be passed to custom_query.
Value

A tibble with the results.

See Also

`geo_lite_sf()`, `tidygeocoder::geo()`

Other geocoding: `geo_address_lookup()`, `geo_amenity()`, `reverse_geo_lite()`

Examples

```r
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
)
```

Description

This function allows you to geocode addresses and return the corresponding spatial object.

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.
Usage

```r
geo_lite_sf(
  address,
  limit = 1,
  return_addresses = TRUE,
  full_results = FALSE,
  verbose = FALSE,
  custom_query = list(),
  points_only = TRUE
)
```

Arguments

- `address`: single line address (i.e. "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. Note that most services return the input addresses with `full_results = TRUE` and setting `return_addresses` to FALSE does not prevent this.
- `full_results`: returns all available data from the geocoding service if TRUE. If FALSE (default) then only latitude and longitude columns are returned from the geocoding service.
- `verbose`: if TRUE then detailed logs are output to the console. FALSE is default. Can be set permanently with `options(tidygeocoder.verbose = TRUE)`
- `custom_query`: API-specific parameters to be used, passed as a named list (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).

Details

See [https://nominatim.org/release-docs/latest/api/Search/](https://nominatim.org/release-docs/latest/api/Search/) for additional parameters to be passed to `custom_query`.

Value

A sf object with the results.

See Also

- `geo_lite()`

Other spatial: `bbox_to_poly()`, `geo_address_lookup_sf()`, `geo_amenity_sf()`, `reverse_geo_lite_sf()`
Examples

```r
# Map - Points
library(ggplot2)
pentagon <- geo_lite_sf("Pentagon")

ggplot(pentagon) +
  geom_sf()
pentagon_poly <- geo_lite_sf("Pentagon", points_only = FALSE)

ggplot(pentagon_poly) +
  geom_sf()

# Several results
Madrid <- geo_lite_sf("Madrid",
                   limit = 2,
                   points_only = FALSE, full_results = TRUE)

ggplot(Madrid) +
  geom_sf(fill = NA)
Starbucks <- geo_lite_sf("Starbucks, New York",
                         limit = 20, full_results = TRUE)

ggplot(Starbucks) +
  geom_sf()
```

---

<table>
<thead>
<tr>
<th>osm_amenities</th>
<th>OpenStreetMap amenity database</th>
</tr>
</thead>
</table>

**Description**

Database with the list of amenities available on OpenStreetMap.

**Format**

A tibble with the amenities and the corresponding category

**Details**
<table>
<thead>
<tr>
<th>category</th>
<th>amenity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sustenance</td>
<td>bar</td>
</tr>
<tr>
<td>Sustenance</td>
<td>biergarten</td>
</tr>
<tr>
<td>Sustenance</td>
<td>cafe</td>
</tr>
<tr>
<td>Sustenance</td>
<td>fast_food</td>
</tr>
<tr>
<td>Sustenance</td>
<td>food_court</td>
</tr>
<tr>
<td>Sustenance</td>
<td>ice_cream</td>
</tr>
<tr>
<td>Sustenance</td>
<td>pub</td>
</tr>
<tr>
<td>Sustenance</td>
<td>restaurant</td>
</tr>
<tr>
<td>Education</td>
<td>college</td>
</tr>
<tr>
<td>Education</td>
<td>driving_school</td>
</tr>
<tr>
<td>Education</td>
<td>kindergarten</td>
</tr>
<tr>
<td>Education</td>
<td>language_school</td>
</tr>
<tr>
<td>Education</td>
<td>library</td>
</tr>
<tr>
<td>Education</td>
<td>toy_library</td>
</tr>
<tr>
<td>Education</td>
<td>music_school</td>
</tr>
<tr>
<td>Education</td>
<td>school</td>
</tr>
<tr>
<td>Education</td>
<td>university</td>
</tr>
<tr>
<td>Transportation</td>
<td>bicycle_parking</td>
</tr>
<tr>
<td>Transportation</td>
<td>bicycle_repair_station</td>
</tr>
<tr>
<td>Transportation</td>
<td>bicycle_rental</td>
</tr>
<tr>
<td>Transportation</td>
<td>boat_rental</td>
</tr>
<tr>
<td>Transportation</td>
<td>boat_sharing</td>
</tr>
<tr>
<td>Transportation</td>
<td>bus_station</td>
</tr>
<tr>
<td>Transportation</td>
<td>car_rental</td>
</tr>
<tr>
<td>Transportation</td>
<td>car_sharing</td>
</tr>
<tr>
<td>Transportation</td>
<td>car_wash</td>
</tr>
<tr>
<td>Transportation</td>
<td>vehicle_inspection</td>
</tr>
<tr>
<td>Transportation</td>
<td>charging_station</td>
</tr>
<tr>
<td>Transportation</td>
<td>ferry_terminal</td>
</tr>
<tr>
<td>Transportation</td>
<td>fuel</td>
</tr>
<tr>
<td>Transportation</td>
<td>grit_bin</td>
</tr>
<tr>
<td>Transportation</td>
<td>motorcycle_parking</td>
</tr>
<tr>
<td>Transportation</td>
<td>parking</td>
</tr>
<tr>
<td>Transportation</td>
<td>parking_entrance</td>
</tr>
<tr>
<td>Transportation</td>
<td>parking_space</td>
</tr>
<tr>
<td>Transportation</td>
<td>taxi</td>
</tr>
<tr>
<td>Financial</td>
<td>atm</td>
</tr>
<tr>
<td>Financial</td>
<td>bank</td>
</tr>
<tr>
<td>Financial</td>
<td>bureau_de_change</td>
</tr>
<tr>
<td>Healthcare</td>
<td>baby_hatch</td>
</tr>
<tr>
<td>Healthcare</td>
<td>clinic</td>
</tr>
<tr>
<td>Healthcare</td>
<td>dentist</td>
</tr>
<tr>
<td>Healthcare</td>
<td>doctors</td>
</tr>
<tr>
<td>Healthcare</td>
<td>hospital</td>
</tr>
<tr>
<td>Healthcare</td>
<td>nursing_home</td>
</tr>
<tr>
<td>Healthcare</td>
<td>pharmacy</td>
</tr>
<tr>
<td>Healthcare</td>
<td>social_facility</td>
</tr>
<tr>
<td>Category</td>
<td>Type</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Healthcare</td>
<td>veterinary</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>arts_centre</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>brothel</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>casino</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>cinema</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>community_centre</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>conference_centre</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>events_venue</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>fountain</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>gambling</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>love_hotel</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>nightclub</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>planetarium</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>public_bookcase</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>social_centre</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>stripclub</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>studio</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>swingerclub</td>
</tr>
<tr>
<td>Entertainment-Arts-Culture</td>
<td>theatre</td>
</tr>
<tr>
<td>Public Service</td>
<td>courthouse</td>
</tr>
<tr>
<td>Public Service</td>
<td>embassy</td>
</tr>
<tr>
<td>Public Service</td>
<td>fire_station</td>
</tr>
<tr>
<td>Public Service</td>
<td>police</td>
</tr>
<tr>
<td>Public Service</td>
<td>post_box</td>
</tr>
<tr>
<td>Public Service</td>
<td>post_depot</td>
</tr>
<tr>
<td>Public Service</td>
<td>post_office</td>
</tr>
<tr>
<td>Public Service</td>
<td>prison</td>
</tr>
<tr>
<td>Public Service</td>
<td>ranger_station</td>
</tr>
<tr>
<td>Public Service</td>
<td>townhall</td>
</tr>
<tr>
<td>Facilities</td>
<td>bbq</td>
</tr>
<tr>
<td>Facilities</td>
<td>bench</td>
</tr>
<tr>
<td>Facilities</td>
<td>dog_toilet</td>
</tr>
<tr>
<td>Facilities</td>
<td>drinking_water</td>
</tr>
<tr>
<td>Facilities</td>
<td>give_box</td>
</tr>
<tr>
<td>Facilities</td>
<td>shelter</td>
</tr>
<tr>
<td>Facilities</td>
<td>shower</td>
</tr>
<tr>
<td>Facilities</td>
<td>telephone</td>
</tr>
<tr>
<td>Facilities</td>
<td>toilets</td>
</tr>
<tr>
<td>Facilities</td>
<td>water_point</td>
</tr>
<tr>
<td>Facilities</td>
<td>watering_place</td>
</tr>
<tr>
<td>Waste Management</td>
<td>sanitary_dump_station</td>
</tr>
<tr>
<td>Waste Management</td>
<td>recycling</td>
</tr>
<tr>
<td>Waste Management</td>
<td>waste_basket</td>
</tr>
<tr>
<td>Waste Management</td>
<td>waste_disposal</td>
</tr>
<tr>
<td>Waste Management</td>
<td>waste_transfer_station</td>
</tr>
<tr>
<td>Others</td>
<td>animal_boarding</td>
</tr>
<tr>
<td>Others</td>
<td>animal_breeding</td>
</tr>
<tr>
<td>Others</td>
<td>animal_shelter</td>
</tr>
</tbody>
</table>
Reverse geocode coordinates

Description
Reverse geocodes geographic coordinates (latitude and longitude) given as numeric values. Latitudes must be between -90 and 90 and longitudes must be between -180 and 180.

Usage
```r
reverse_geo_lite(
    lat,
    long,
    address = "address",
    full_results = FALSE,
    return_coords = TRUE,
    verbose = FALSE,
    custom_query = list()
)
```
reverse_geo_lite

Arguments

- **lat**: latitude values (input data)
- **long**: longitude values (input data)
- **address**: name of the address column (in the output data)
- **full_results**: returns all available data from the geocoding service if TRUE. If FALSE (default) then only a single address column is returned from the geocoding service.
- **return_coords**: return input coordinates with results if TRUE. Note that most services return the input coordinates with `full_results = TRUE` and setting `return_coords` to FALSE does not prevent this.
- **verbose**: if TRUE then detailed logs are output to the console. FALSE is default. Can be set permanently with `options(tidygeocoder.verbose = TRUE)`
- **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/develop/api/Reverse/](https://nominatim.org/release-docs/develop/api/Reverse/) for additional parameters to be passed to `custom_query`.

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

<table>
<thead>
<tr>
<th>zoom</th>
<th>address_detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>country</td>
</tr>
<tr>
<td>5</td>
<td>state</td>
</tr>
<tr>
<td>8</td>
<td>county</td>
</tr>
<tr>
<td>10</td>
<td>city</td>
</tr>
<tr>
<td>14</td>
<td>suburb</td>
</tr>
<tr>
<td>16</td>
<td>major streets</td>
</tr>
<tr>
<td>17</td>
<td>major and minor streets</td>
</tr>
<tr>
<td>18</td>
<td>building</td>
</tr>
</tbody>
</table>

Value

A tibble with the results.

See Also

- `reverse_geo_lite_sf()`, `tidygeocoder::reverse_geo()
- Other geocoding: `geo_address_lookup()`, `geo_amenity()`, `geo_lite()`

Examples
**reverse_geo_lite_sf**

Get spatial objects through reverse geocoding

**Description**

This function allows you to extract the spatial object located on a known pair of coordinates (lat, long). Latitudes must be between -90 and 90 and longitudes must be between -180 and 180.

**Usage**

```r
reverse_geo_lite_sf(
  lat,  # latitude values (input data)
  long,  # longitude values (input data)
  address = "address",  # name of the address column (in the output data)
  full_results = FALSE,  # returns all available data from the geocoding service if TRUE. If FALSE (default) then only a single address column is returned from the geocoding service.
  return_coords = TRUE,
  verbose = FALSE,
  custom_query = list(),
  points_only = TRUE
)
```

**Arguments**

- `lat` : latitude values (input data)
- `long` : longitude values (input data)
- `address` : name of the address column (in the output data)
- `full_results` : returns all available data from the geocoding service if TRUE. If FALSE (default) then only a single address column is returned from the geocoding service.
return_coords  return input coordinates with results if TRUE. Note that most services return the input coordinates with full_results = TRUE and setting return_coords to FALSE does not prevent this.

verbose  if TRUE then detailed logs are output to the console. FALSE is default. Can be set permanently with options(tidygeocoder.verbose = TRUE)

custom_query  API-specific parameters to be used, passed as a named list (ie. list(zoom = 3)). 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).

Details

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

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

<table>
<thead>
<tr>
<th>zoom</th>
<th>address_detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>country</td>
</tr>
<tr>
<td>5</td>
<td>state</td>
</tr>
<tr>
<td>8</td>
<td>county</td>
</tr>
<tr>
<td>10</td>
<td>city</td>
</tr>
<tr>
<td>14</td>
<td>suburb</td>
</tr>
<tr>
<td>16</td>
<td>major streets</td>
</tr>
<tr>
<td>17</td>
<td>major and minor streets</td>
</tr>
<tr>
<td>18</td>
<td>building</td>
</tr>
</tbody>
</table>

Value

A sf object with the results.

See Also

reverse_geo_lite()

Other spatial: bbox_to_poly(), geo_address_lookup_sf(), geo_amenity_sf(), geo_lite_sf()

Examples

library(ggplot2)

Coliseum <- geo_lite("Coliseo, Rome, Italy")

# Coliseum
Col_sf <- reverse_geo_lite_sf(
  lat = Coliseum$lat,
reverse_geo_lite_sf

lon = Coliseum$lon,
points_only = FALSE
}
ggplot(Col.sf) +
  geom_sf()

# City of Rome - Zoom 10
Rome.sf <- reverse_geo_lite_sf(
  lat = Coliseum$lat,
  lon = Coliseum$lon,
  custom_query = list(zoom = 10),
  points_only = FALSE
)
ggplot(Rome.sf) +
  geom_sf()

# County - Zoom 8
County.sf <- reverse_geo_lite_sf(
  lat = Coliseum$lat,
  lon = Coliseum$lon,
  custom_query = list(zoom = 8),
  points_only = FALSE
)
ggplot(County.sf) +
  geom_sf()
Index

* amenity
  bbox_to_poly, 2
  geo_amenity, 7
  geo_amenity_sf, 9
  osm_amenities, 14

* datasets
  osm_amenities, 14

* geocoding
  geo_address_lookup, 3
  geo_amenity, 7
  geo_lite, 11
  reverse_geo_lite, 17

* lookup
  geo_address_lookup, 3
  geo_address_lookup_sf, 5

* spatial
  bbox_to_poly, 2
  geo_address_lookup_sf, 5
  geo_amenity_sf, 9
  geo_lite_sf, 12
  reverse_geo_lite_sf, 19
  bbox_to_poly, 2, 6, 8, 10, 13, 17, 20
  geo_address_lookup, 3, 6, 8, 12, 18
  geo_address_lookup_sf, 3, 4, 5, 10, 13, 20
  geo_amenity, 3, 4, 7, 10, 12, 17, 18
  geo_amenity_sf, 3, 6, 8, 9, 13, 17, 20
  geo_lite, 4, 8, 11, 18
  geo_lite(), 7, 9, 13
  geo_lite_sf, 3, 6, 10, 12, 20
  geo_lite_sf(), 12
  osm_amenities, 3, 7–10, 14
  reverse_geo_lite, 4, 8, 12, 17
  reverse_geo_lite(), 20
  reverse_geo_lite_sf, 3, 6, 10, 13, 19
  reverse_geo_lite_sf(), 18
  sf::st_as_sfc(), 3

st_crs, 2

tidygeocoder::geo(), 12

tidygeocoder::reverse_geo(), 18