Package 'nominatimlite'

March 1, 2024

Type Package

Title Interface with 'Nominatim' API Service

Version 0.3.0

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), lifecycle, sf (>= 0.9.0), utils

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

VignetteBuilder knitr

Config/Needs/website dieghernan/gitdevr

Config/testthat/edition 3

Config/testthat/parallel true

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

Encoding UTF-8

RoxygenNote 7.3.1

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] (<https://orcid.org/0000-0002-0375-5156>)

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

Repository CRAN

Date/Publication 2024-03-01 19:30:02 UTC

R topics documented:

`bbox_to_poly` *Coerce a bounding box to a* `sfc POLYGON` *object*

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 `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 `sfc` object of class `POLYGON`.

See Also

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

Get **sf** objects: geo_address_lookup_sf(), geo_lite_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 for OSM elements*

Description

The lookup API allows to query the address and other details of one or multiple OSM objects like node, way or relation. This function 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,
  custom_query = list()
)
```

Arguments

osm_ids	vector of OSM identifiers as **numeric** (c(00000, 11111, 22222)).
type	vector character 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 in the output data (default "lat").
long	longitude column name in the output data (default "long").
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.
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/develop/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 strings: geo_address_lookup_sf(), geo_lite(), geo_lite_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 for OSM elements in* R*hrefhttps://CRAN.R-project.org/package=sf***sf** *format*

Description

The lookup API allows to query the address and other details of one or multiple OSM objects like node, way or relation. This function 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,
  custom_query = list(),
  points_only = TRUE
)
```

Arguments

`osm_ids`	vector of OSM identifiers as **numeric** (`c(00000, 11111, 22222)`).
`type`	vector character 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 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.

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 strings: geo_address_lookup(), geo_lite(), geo_lite_sf()

Get **sf** objects: bbox_to_poly(), geo_lite_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_lite` *Address search API for OSM elements*

Description

Geocodes addresses given as character values. This function returns the **tibble** associated with the query, see `geo_lite_sf()` for retrieving the data as a spatial object (**sf** format).

Usage

```
geo_lite(
  address,
  lat = "lat",
  long = "lon",
  limit = 1,
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  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.
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.

See Also

geo_lite_sf(), tidygeocoder::geo()

Geocoding strings: geo_address_lookup(), geo_address_lookup_sf(), geo_lite_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 *Address search API for OSM elements in* R*hrefhttps://CRAN.R-project.org/package=sf***sf** *format*

Description

This function allows you to geocode addresses and return the corresponding spatial object. This function returns the spatial object associated with the query using **sf**, see `geo_lite_sf()` for retrieving the data in **tibble** format.

Usage

```
geo_lite_sf(
  address,
  limit = 1,
  return_addresses = TRUE,
  full_results = FALSE,
  verbose = FALSE,
  progressbar = TRUE,
  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.
`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

Geocoding strings: `geo_address_lookup()`, `geo_address_lookup_sf()`, `geo_lite()`

Get sf objects: `bbox_to_poly()`, `geo_address_lookup_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("Madrid",
  limit = 2,
  points_only = FALSE, full_results = TRUE
)
```

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

`reverse_geo_lite` *Reverse geocoding API for OSM elements*

Description

Generates an address from a latitude and longitude. Latitudes must be between `[-90, 90]` and longitudes between `[-180, 180]`. This function 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,
  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.
`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/develop/api/Reverse/` for additional parameters to be passed to `custom_query`.

Value

A **tibble** 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

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 = 1),
  verbose = TRUE, full_results = TRUE
)

dplyr::glimpse(sev)
```

reverse_geo_lite_sf *Reverse geocoding API for OSM elements in* R*hrefhttps://CRAN.R-project.org/package=sf***sf** *format*

Description

Generates an address from a latitude and longitude. Latitudes must be between `[-90, 90]` and longitudes between `[-180, 180]`. This function 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,
  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 `return_addresses`.
`return_coords`	return input coordinates with results if `TRUE`.
`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.
`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`). See **About Geometry Types**.

Details

See `https://nominatim.org/release-docs/develop/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_lite_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