

Package: nlgeocoder (via r-universe)

June 2, 2026

Title Geocoding for the Netherlands

Version 0.2.1

Description R interface to the open location server API of 'Publieke Diensten Op de Kaart' (<<http://www.pdok.nl>>). It offers geocoding, address suggestions and lookup of geographical objects. Included is an utility function for displaying leaflet tiles restricted to the Netherlands.

Depends R (>= 3.4)

License GPL-2

BugReports <https://github.com/uRosConf/nlgeocoder/issues>

URL <https://github.com/uRosConf/nlgeocoder>,
<https://urosconf.github.io/nlgeocoder/>

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Suggests testthat, knitr, rmarkdown, sf, shiny, leaflet

VignetteBuilder knitr

Imports jsonlite

Roxygen list(markdown = TRUE)

Repository <https://urosconf.r-universe.dev>

Date/Publication 2025-12-04 15:06:12 UTC

RemoteUrl <https://github.com/urosconf/nlgeocoder>

RemoteRef HEAD

RemoteSha 60d3f9888a39fdcfb7f73eb7e46fdb63bb41e2d4

Contents

nlgeocoder-package	2
addPdokTiles	3
addresses	4
nl_free	4
nl_geocode	5
nl_lookup	7
nl_reverse	8
nl_suggest	8

Index	10
--------------	-----------

nlgeocoder-package	<i>nlgeocoder package</i>
--------------------	---------------------------

Description

The nlgeocoder package provides a R wrapper around the pdok webservice.

Details

it allows to geocode data using the following functions:

- `nl_geocode()`: geocode a vector of addresses.
- `nl_free()`: find possible matches for one address.
- `nl_suggest()`: find possible suggestions for one address.
- `nl_lookup()`: lookup information of object found in `nl_suggest`.

Author(s)

Maintainer: Edwin de Jonge <edwindjonge@gmail.com> ([ORCID](#))

Authors:

- Willy Tadema
- Egge-Jan Pollé

Other contributors:

- Juris Breidaks [contributor]

See Also

Useful links:

- <https://github.com/uRosConf/nlgeocoder>
- <https://uroscnf.github.io/nlgeocoder/>
- Report bugs at <https://github.com/uRosConf/nlgeocoder/issues>

addPdokTiles	<i>Leaflet function to add pdok tiles</i>
--------------	---

Description

This function adds PDOK tile layers to `leaflet::leaflet()`. It is a small wrapper around `leaflet::addTiles()` that sets the correct urls for the layers.

Usage

```
addPdokTiles(  
  map,  
  type = c("brt", "aerial", "pastel", "gray"),  
  group = match.arg(type),  
  options,  
  ...  
)
```

Arguments

map	leaflet object
type	one of the PDOK tiles: currently: "brt", "aerial", "pastel" or "gray"
group	group is set equal to type
options	passed to <code>leaflet::addTiles()</code>
...	Passed to <code>leaflet::addTiles()</code> .

Details

PDOK provides tiles that can be used as a tile layer in several geovisualization tools including leaflet. The layers differ from other tile layers such as Openstreetmap, Google and cartomap in the following manner:

- The tiles are restricted to the Netherlands: tiles accross the Dutch border are "grayed out". This is useful for cartographic applications that are restricted to Dutch geo-information.
- The tiles are open source and based on open sourced geo-information.

 addresses

Several Points of Interest (POI) that are included for test purposes.

Description

- POI, points of interest
- Address, Uncleaned address information.

Usage

```
addresses
```

Format

An object of class `data.frame` with 5 rows and 2 columns.

 nl_free

Free geocoding search

Description

This function wraps the "free" pdok service, and allows for free search. For syntax and examples see the documentation of `pdok`. A more easy/convenient but simpler function for geocoding is [nl_geocode\(\)](#).

Usage

```
nl_free(
  q,
  rows = NULL,
  start = NULL,
  fq = NULL,
  lat = NULL,
  lon = NULL,
  type = NULL,
  fl = NULL,
  df = NULL,
  ...,
  verbose = FALSE
)
```

Arguments

q	query to geocoding service.
rows	maximum number of rows to be returned. Default is 10.
start	start index. Default is 0.
fq	apply a filter to the query, e.g. fq=bron:BAG. The default filter is 'type:(municipality OR town OR neighborhood OR road OR postcode OR address)'. Use fq=* to remove the default filter.
lat	enter a decimal degree (latitude, in WGS84). Together with lon this will define a point to search from. Search results will be ordered by distance from this point. For example: 'lat=52.09&lon=5.12'
lon	enter a decimal degree (longitude, in WGS84). Together with lat this will define point to search from. Search results will be ordered by distance from this point. For example: 'lat=52.09&lon=5.12'
type	restrict the results on a specific type.
fl	fields to return.
df	field that should be search in.
...	parameters passed to the pdok webservice
verbose	logical should the function print out messages.

Value

The result of the pdok free webservice converted to a R list object.

See Also

[nl_geocode\(\)](#)

Examples

```
if (requireNamespace("sf", quietly = TRUE)){
  l <- nl_free("Henri Faasdreef 312")
  l$response$numFound
  l$response$docs["weergavenaam"]
}
```

nl_geocode

Geocode adresses

Description

nl_geocode returns for a vector of addresses the most probable object/location. This function is more user friendly than the barebones webservices ([nl_free\(\)](#)), and uses the same function signature as `ggmap::geocode`.

Usage

```

nl_geocode(
  location,
  output = c("wgs84", "rd", "data.frame"),
  messaging = FALSE,
  type = "adres",
  ...,
  verbose = messaging
)

nl_geocode_rd(
  location,
  messaging = FALSE,
  type = "adres",
  ...,
  verbose = messaging
)

nl_geocode_df(
  location,
  messaging = FALSE,
  type = "adres",
  ...,
  verbose = messaging
)

```

Arguments

location	string with location to be found
output	Should the output be a <code>data.frame()</code> or <code>sf::sf()</code> object in wgs84 or Rijksdriehoekstelsel format?
messaging	Print the urls fired to the webserver (consistent with <code>ggmap::geocode</code>)
type	restrict the type of object that is returned from the service, see details for possible types.
...	will be passed to <code>nl_free()</code> .
verbose	identical to <code>messaging</code> (consistent with other <code>nlgeoder</code> functions)

Details

type can be one or more of the following: "provincie", "gemeente", "woonplaats", "weg", "post-code", "adres", "perceel", "hectometerpaal", "wijk", "buurt", "waterschapsgrens", "appartementrecht".

Value

The return type can be specified and can be of type "sf" or "data.frame", depending on the value of output.

Examples

```
data("addresses")
r <- nl_geocode(addresses$Address, output = "data.frame")
r[["weergavenaam"]]
names(r)

# or if you have sf installed
if (requireNamespace("sf", quietly = TRUE)){
  r_sf <- nl_geocode(addresses$Address, output = "wgs84")
  print(r_sf)
}
```

nl_lookup

Look up a geo object

Description

Retrieve detailed properties of a geo object found with [nl_suggest\(\)](#) or [nl_free\(\)](#).

Usage

```
nl_lookup(id, ..., output = c("list", "raw"), verbose = FALSE)
```

Arguments

id	of object found in nl_suggest or nl_free
...	extra parameters are passed to the lookup service of pdok .
output	What type of output should be returned
verbose	should the function print message while retrieving the data?

Value

Depending on the value of `output` the raw search results in R format or the properties of the specific object as a R list object.

Examples

```
obj <- nl_lookup("weg-f633e85f07eda4e68a00fb13f9d128f5", output = "list")
names(obj)
obj$weergavenaam
```

nl_reverse	<i>experimental reverse api</i>
------------	---------------------------------

Description

experimental reverse api

Usage

```
nl_reverse()
```

nl_suggest	<i>Get a list of suggestions for geolocations in NL</i>
------------	---

Description

nl_suggest returns a list of suggestions for a location description.

Usage

```
nl_suggest(
  q,
  ...,
  rows = NULL,
  type = NULL,
  verbose = FALSE,
  fl = NULL,
  sort = NULL,
  qf = NULL,
  bq = NULL
)
```

Arguments

q	search terms that should be geolocated
...	parameters passed to geolocation service
rows	maximum number of rows to be returned. Default is 10.
type	restrict type of geolocation to a type (see details for possible types)
verbose	Should the functions print messages on what it is retrieving.
fl	the columns that should be returned (aka select on columns of result)
sort	how the data should be sorted
qf	the fields that should be queried
bq	the boosting of the query.

Details

type can be one or more of the following: "provincie", "gemeente" , "woonplaats", "weg", "post-code", "adres", "perceel", "hectometerpaal", "wijk", "buurt", "waterschapsgrens", "appartementsrecht".

Value

The result of the pdok suggest webservice converted to a R list object.

Examples

```
sug <- nl_suggest("Henri Faasdreef")

# how many objects have a score?
sug$response$numFound

# get suggestions
sug$response$docs
```

Index

* datasets

addresses, 4

addPdokTiles, 3

addresses, 4

data.frame(), 6

leaflet::addTiles(), 3

leaflet::leaflet(), 3

nl_free, 4

nl_free(), 2, 5–7

nl_geocode, 5

nl_geocode(), 2, 4, 5

nl_geocode_df (nl_geocode), 5

nl_geocode_rd (nl_geocode), 5

nl_lookup, 7

nl_lookup(), 2

nl_reverse, 8

nl_suggest, 8

nl_suggest(), 2, 7

nlgeocoder (nlgeocoder-package), 2

nlgeocoder-package, 2

sf::sf(), 6