Package ‘mapsapi’

September 7, 2019

Type Package

Title 'sf'-Compatible Interface to 'Google Maps' APIs

Version 0.4.2

Description Interface to the 'Google Maps' APIs: (1) routing directions based on the 'Directions' API, returned as 'sf' objects, either as single feature per alternative route, or a single feature per segment per alternative route; (2) travel distance or time matrices based on the 'Distance Matrix' API; (3) geocoded locations based on the 'Geocode' API, returned as 'sf' objects, either points or bounds.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports magrittr (>= 1.5), xml2 (>= 1.1.1), sf (>= 0.5-3), bitops (>= 1.0-6), plyr (>= 1.8.4)

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, leaflet

VignetteBuilder knitr

URL https://github.com/michaeldorman/mapsapi/

BugReports https://github.com/michaeldorman/mapsapi/issues/

NeedsCompilation no

Author Michael Dorman [aut, cre],
    Tom Buckley [ctb],
    Alex Dannenberg [ctb],
    Mihir Bhaskar [ctb]

Maintainer Michael Dorman <dorman@post.bgu.ac.il>

Repository CRAN

Date/Publication 2019-09-07 15:50:02 UTC
R topics documented:

mp_directions .............................................................. 2
mp_geocode ................................................................. 4
mp_get_bounds .............................................................. 6
mp_get_matrix .............................................................. 6
mp_get_points .............................................................. 8
mp_get_routes .............................................................. 8
mp_get_segments ........................................................... 10
mp_matrix ................................................................. 11
response_directions_driving ........................................... 12
response_directions_transit ............................................. 13
response_geocode ......................................................... 14
response_matrix ........................................................... 14

Index 15

mp_directions  Get directions from the Google Maps Directions API

Description

Get directions from the Google Maps Directions API

Usage

mp_directions(origin, waypoints = NULL, destination,
mode = c("driving", "transit", "walking", "bicycling"),
arrival_time = NULL, departure_time = NULL, alternatives = FALSE,
avoid = NULL, region = NULL, key = NULL, quiet = FALSE)

Arguments

origin  Origin, as
• character vector of length one with address to be geocoded
• numeric vector of length two (lon, lat)
• matrix with one row and two columns (lon, lat)
• sf or sfc point layer with one feature

waypoints  Waypoints, in one of the same formats as for origins but possibly with more
than one location, i.e.
• character vector with addresses to be geocoded
• numeric vector of length two (lon, lat)
• matrix with two columns (lon, lat)
• sf or sfc point layer

destination  Destination, in one of the same formats as for origins

mode  Travel mode, one of: "driving" (default), "transit", "walking", "bicycling"
arrival_time  The desired time of arrival for transit directions, as POSIXct
departure_time The desired time of departure, as POSIXct
alternatives  Whether to return more than one alternative (logical)
avoid        NULL (default) or one of: "tolls", "highways", "ferries" or "indoor"
region       The region code, specified as a ccTLD ("top-level domain") two-character value
              (e.g. "es" for Spain) (optional)
key          Google APIs key (optional)
quiet        Logical; suppress printing URL for Google Maps API call (e.g. to hide API key)

Value
XML document with Google Maps Directions API response

Note
- Use function mp_get_routes to extract sf line layer where each feature is a route
- Use function mp_get_segments to extract sf line layer where each feature is a route segment

References
https://developers.google.com/maps/documentation/directions/intro

Examples
# Built-in response example
library(xml2)
doc = as_xml_document(response_directions_driving)
r = mp_get_routes(doc)
seg = mp_get_segments(doc)

## Not run:
# Text file with API key
key = readLines("~/key")

# Using 'numeric' input
doc = mp_directions(
    origin = c(34.81127, 31.89277),
    destination = c(34.781107, 32.085003),
    alternatives = TRUE,
    key = key
)

# Using 'character' and 'sf' input
library(sf)
doc = mp_directions(
    origin = "Beer-Sheva",
    destination = c(34.781107, 32.085003) %>% st_point %>% st_sfc(crs = 4326),
    alternatives = TRUE,
### mp_geocode

Get geocoded coordinates using the Google Maps Geocoding API

#### Description

Get geocoded coordinates using the Google Maps Geocoding API

#### Usage

```r
mp_geocode(addresses, region = NULL, postcode = NULL, bounds = NULL,
           key = NULL)
```

#### Arguments

- **addresses**: Addresses to geocode, as character vector
- **region**: The region code, specified as a ccTLD (“top-level domain”) two-character value (e.g. "es" for Spain). This can be a character vector of length 1 (in which case it is replicated) or a character vector with the same length of addresses (optional)
- **postcode**: Vector of postal codes to filter the address match by (optional); Note that this is a component filter, which means that for each address, Google will search only within the corresponding postal code if nonmissing
- **bounds**: A preferred bounding box, specified as a numeric vector with four values xmin/ymin/xmax/ymax (in latitude/longitude) representing the coordinates of the southwest and northeast corners, e.g. as returned by function ‘sf::st_bbox’. This can be a single vector (in which case it is replicated) or a list of numeric vectors with the same length as addresses (optional)
- **key**: Google APIs key (optional)

#### Value

list of XML documents with Google Maps Geocoding API responses, one item per element in addresses

#### Note

- Use function `mp_get_points` to extract locations as sf point layer
- Use function `mp_get_bounds` to extract location bounds as sf polygonal layer

#### References

[https://developers.google.com/maps/documentation/geocoding/intro](https://developers.google.com/maps/documentation/geocoding/intro)
Examples

```r
# Built-in response example
library(xml2)
doc = list("Tel-Aviv" = as_xml_document(response_geocode))
pnt = mp_get_points(doc)
bounds = mp_get_bounds(doc)

## Not run:

# Text file with API key
key = readLines("~/key")

# Basic use
addresses = c("Rehovot", "Beer-Sheva", "New-York")
doc = mp_geocode(addresses)  # without key
doc = mp_geocode(addresses, key = key)  # with key
pnt = mp_get_points(doc)
pnt

# Using the 'region' parameter
doc = mp_geocode(addresses = "Toledo", key = key)
mp_get_points(doc)
doc = mp_geocode(addresses = "Toledo", region = "es", key = key)
mp_get_points(doc)

# Various addresses
addresses = c(
  "Baker Street 221b, London",
  "Brandenburger Tor, Berlin",
  "",
  "Platz der Deutschen Einheit 1, Hamburg",
  "Arc de Triomphe de l'Etoile, Paris",
  NA
)
doc = mp_geocode(addresses, key = key)
pnt = mp_get_points(doc)
pnt

# Specifying a bounding box
b = c(-118.604794, 34.172684, -118.500938, 34.236144)  # Bounds as xmin/ymin/xmax/ymax
result = mp_geocode(addresses = "Winnetka", key = key)
mp_get_points(result)
result = mp_geocode(addresses = "Winnetka", bounds = b, key = key)
mp_get_points(result)
result = mp_geocode(addresses = rep("Winnetka", 3), bounds = list(b, NA, b), key = key)
mp_get_points(result)

## End(Not run)
```
**mp_get_bounds**

*Extract geocoded *bounds* from Google Maps Geocode API response*

**Description**

Extract geocoded *bounds* from Google Maps Geocode API response

**Usage**

```r
mp_get_bounds(doc)
```

**Arguments**

- **doc**
  XML document with Google Maps Geocode API response

**Value**

- **sf**
  Polygonal layer representing bounds of geocoded locations

**Examples**

```r
library(xml2)
doc = list("Tel-Aviv" = as_xml_document(response_geocode))
b = mp_get_bounds(doc)
## Not run:
doc = mp_geocode(addresses = c("Tel-Aviv", "Rehovot", "Beer-Sheva"))
b = mp_get_bounds(doc)
## End(Not run)
```

---

**mp_get_matrix**

*Extract distance or duration *matrix* from a Google Maps Distance Matrix API response*

**Description**

Extract distance or duration *matrix* from a Google Maps Distance Matrix API response

**Usage**

```r
mp_get_matrix(doc, value = "distance_m")
```

**Arguments**

- **doc**
  XML document with Google Maps Distance Matrix API response
- **value**
  Value to extract, one of: "distance_m" (default), "distance_text", "duration_s", "duration_text", "duration_in_traffic_s", "duration_in_traffic_text"
Value

A matrix, where rows represent origins and columns represent destinations. Matrix values are according to selected value, or NA if the ARI returned zero results.

Note

The "duration_in_traffic_s" and "duration_in_traffic_text" options are only applicable when the API response contains these fields, i.e., when using `mp_matrix` with `mode="driving"`, with `departure_time` specified, and API key `key` provided.

Examples

```r
library(xml2)
doc = as_xml_document(response_matrix)
mp_get_matrix(doc, value = "distance_m")
mp_get_matrix(doc, value = "distance_text")
mp_get_matrix(doc, value = "duration_s")
mp_get_matrix(doc, value = "duration_text")

## Not run:
key = readLines("~/key") # Text file with API key
locations = c("Tel-Aviv", "Jerusalem", "Neve Shalom")

# Driving times
doc = mp_matrix(
  origins = locations,
  destinations = locations,
  mode = "driving",
  departure_time = Sys.time() + as.difftime(10, units = "mins"),
  key = key
)
mp_get_matrix(doc, value = "distance_m")
mp_get_matrix(doc, value = "distance_text")
mp_get_matrix(doc, value = "duration_s")
mp_get_matrix(doc, value = "duration_text")
mp_get_matrix(doc, value = "duration_in_traffic_s")
mp_get_matrix(doc, value = "duration_in_traffic_text")

# Public transport times
doc = mp_matrix(
  origins = locations,
  destinations = locations,
  mode = "transit",
  key = key
)
mp_get_matrix(doc, value = "distance_m")
mp_get_matrix(doc, value = "distance_text")
mp_get_matrix(doc, value = "duration_s")
mp_get_matrix(doc, value = "duration_text")
```
mp_get_points

**Extract geocoded points from Google Maps Geocode API response**

**Description**

Extract geocoded points from Google Maps Geocode API response

**Usage**

```r
mp_get_points(doc, all_results = FALSE)
```

**Arguments**

- `doc`: XML document with Google Maps Geocode API response
- `all_results`: The geocoder may return several results when address queries are ambiguous. Should all results be returned (TRUE), or just the first one (FALSE, default)?

**Value**

- `sf`: Point layer representing geocoded locations

**Examples**

```r
library(xml2)
doc = list("Tel-Aviv" = as_xml_document(response_geocode))
pnt = mp_get_points(doc)
```

## End(Not run)

mp_get_routes

**Extract *routes* from Google Maps Directions API response**

**Description**

Extract *routes* from Google Maps Directions API response

**Usage**

```r
mp_get_routes(doc)
```
**mp_get_routes**

**Arguments**

- **doc**: XML document with Google Maps Directions API response

**Value**

Line layer (class `sf`) representing routes.

When document contains no routes ("ZERO_RESULTS" status), the function returns an empty line layer with `NA` in all fields.

**Examples**

```r
library(xml2)

doc = as_xml_document(response_directions_driving)
r = mp_get_routes(doc)
plot(r)

doc = as_xml_document(response_directions_transit)
r = mp_get_routes(doc)
plot(r)

## Not run:
# Text file with API key
key = readLines("~/key")

# Transit example
doc = mp_directions(
  origin = c(34.81127, 31.89277),
  destination = c(34.781107, 32.085003),
  mode = "transit",
  alternatives = TRUE,
  key = key
)
r = mp_get_routes(doc)
plot(r)

# Duration in traffic
doc = mp_directions(
  origin = c(34.81127, 31.89277),
  destination = c(34.781107, 32.085003),
  departure_time = Sys.time(),
  alternatives = TRUE,
  key = key
)
r = mp_get_routes(doc)
plot(r)

# Using waypoints
doc = mp_directions(
  origin = c(34.81127, 31.89277),
  destination = c(34.781107, 32.085003),
  alternatives = TRUE,
  key = key
)
r = mp_get_routes(doc)
plot(r)
```
```r
waypoints = rbind(c(35.01582, 31.90020), c(34.84246, 31.85356)),
destination = c(34.781107, 32.085003),
key = key
)
r = mp_get_routes(doc)
plot(r)

## End(Not run)
```

---

**mp_get_segments**

*Extract route segments* from a Google Maps Directions API response

**Description**

Extract *route segments* from a Google Maps Directions API response

**Usage**

```
mp_get_segments(doc)
```

**Arguments**

- `doc` XML document with Google Maps Directions API response

**Value**

Line layer (class `sf`) representing route segments

**Examples**

```r
library(xml2)

doc = as_xml_document(response_directions_driving)
seg = mp_get_segments(doc)
plot(seg)

doc = as_xml_document(response_directions_transit)
seg = mp_get_segments(doc)
plot(seg)

## Not run:

# Text file with API key
key = readLines("~/key")

# Transit example
doc = mp_directions(
```
```r
origin = c(34.81127, 31.89277),
destination = c(34.781107, 32.085003),
mode = "transit",
alternatives = TRUE,
key = key
)
seg = mp_get_segments(doc)
plot(seg)

# Using waypoints
doc = mp_directions(
  origin = c(34.81127, 31.89277),
  waypoints = rbind(c(35.01582, 31.90020), c(34.84246, 31.85356)),
destination = c(34.781107, 32.085003),
  alternatives = TRUE,
  key = key
)
seg = mp_get_segments(doc)
plot(seg)

## End(Not run)
```

---

**mp_matrix**

*Get distance matrix from the Google Maps Distance Matrix API*

**Description**

Get distance matrix from the Google Maps Distance Matrix API

**Usage**

```r
mp_matrix(origins, destinations, mode = c("driving", "transit",
  "walking", "bicycling"), arrival_time = NULL, departure_time = NULL,
  avoid = NULL, region = NULL, key = NULL, quiet = FALSE)
```

**Arguments**

- **origins**: Origins, as
  - character vector with addresses to be geocoded
  - numeric vector of length two (lon, lat)
  - matrix with two columns (lon, lat)
  - sf or sfc point layer
- **destinations**: Destinations, in one of the same formats as for origins
- **mode**: Travel mode, one of: "driving", "transit", "walking", "bicycling"
- **arrival_time**: The desired time of arrival for transit directions, as POSIXct
- **departure_time**: The desired time of departure, as POSIXct
response_directions_driving

| avoid | NULL (default) or one of: "tolls", "highways", "ferries" or "indoor" |
| region | The region code, specified as a ccTLD ("top-level domain") two-character value (e.g. "es" for Spain) (optional) |
| key | Google APIs key (optional) |
| quiet | Logical; suppress printing URL for Google Maps API call (e.g. to hide API key) |

**Value**

XML document with Google Maps Distance Matrix API response

**Note**

Use function `mp_get_matrix` to extract **distance** and **duration** matrix objects

**References**

https://developers.google.com/maps/documentation/distance-matrix/intro

**Examples**

```r
# Built-in response example
library(xml2)
doc = as_xml_document(response_matrix)

## Not run:
# Using 'data.frame' input
doc = mp_matrix(
  origins = rbind(c(34.81127, 31.89277), c(35.212085, 31.769976)),
  destinations = c(34.781107, 32.085003)
)

# Using 'character' input
locations = c("Haifa", "Tel-Aviv", "Jerusalem", "Beer-Sheva")
doc = mp_matrix(
  origins = locations,
  destinations = locations
)

## End(Not run)
```

---

**response_directions_driving**

*Sample response from Google Maps Directions API*

**Description**

XML documents with **driving** directions from Tel-Aviv to Haifa
**Usage**

`response_directions_driving`

**Format**

A character vector of length one

**Note**

See `response_directions_transit` for Directions API response with `transit` directions

**Examples**

```r
define_library(xml2)
doc <- as_xml_document(response_directions_driving)
```

---

**Description**

XML documents with `transit` directions from New-York to Boston

**Usage**

`response_directions_transit`

**Format**

A character vector of length one

**Note**

See `response_directions_driving` for Directions API response with `driving` directions

**Examples**

```r
define_library(xml2)
doc <- as_xml_document(response_directions_transit)
```
**response_geocode**

*Sample response from Google Maps Geocode API*

**Description**

An XML document with a geocoded location for the address "Tel-Aviv"

**Usage**

`response_geocode`

**Format**

A character vector of length one

**Examples**

```r
library(xml2)
doc = as_xml_document(response_geocode)
```

---

**response_matrix**

*Sample response from Google Maps Distance Matrix API*

**Description**

An XML document with a distance matrix for driving between three locations: Tel-Aviv, Jerusalem and Beer-Sheva

**Usage**

`response_matrix`

**Format**

A character vector of length one

**Examples**

```r
library(xml2)
doc = as_xml_document(response_matrix)
```
Index

*Topic datasets
  response_directions_driving, 12
  response_directions_transit, 13
  response_geocode, 14
  response_matrix, 14

mp_directions, 2
mp_geocode, 4
mp_get_bounds, 4, 6
mp_get_matrix, 6, 12
mp_get_points, 4, 8
mp_get_routes, 3, 8
mp_get_segments, 3, 10
mp_matrix, 7, 11

response_directions_driving, 12, 13
response_directions_transit, 13, 13
response_geocode, 14
response_matrix, 14