Package ‘mapsapi’

July 21, 2023

Type Package
Title 'sf'-Compatible Interface to 'Google Maps' APIs
Version 0.5.4
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; (4) map images using the 'Maps Static' API, returned as 'stars' objects.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Depends R (>= 4.1.0)
Imports xml2, sf, bitops, stars, RgoogleMaps, httr
RoxygenNote 7.2.3
Suggests knitr, rmarkdown, leaflet, ggplot2, dplyr
VignetteBuilder knitr
BugReports https://github.com/michaeldorman/mapsapi/issues/
NeedsCompilation no
Author Michael Dorman [aut, cre],
    Tom Buckley [ctb],
    Alex Dannenberg [ctb],
    Mihir Bhaskar [ctb],
    Juan P. Fonseca-Zamora [ctb],
    Rodrigo Vega [ctb]
Maintainer Michael Dorman <dorman@post.bgu.ac.il>
Repository CRAN
Date/Publication 2023-07-21 12:10:02 UTC
**R topics documented:**

mp_directions .................................................. 2
mp_geocode ......................................................... 5
mp_get_bounds ..................................................... 7
mp_get_matrix ...................................................... 8
mp_get_points ..................................................... 9
mp_get_routes ..................................................... 10
mp_get_segments .................................................. 11
mp_map .............................................................. 13
mp_matrix .......................................................... 15
plot.mapsapi_map .................................................. 17
response_directions_driving ..................................... 18
response_directions_transit ..................................... 19
response_geocode .................................................. 19
response_map ....................................................... 20
response_matrix ................................................... 20

**Index**

mp_directions ........................ Get directions from the Google Maps Directions API

**Description**

Get directions from the Google Maps Directions API

**Usage**

```r
mp_directions(
  origin,
  waypoints = NULL,
  destination,
  mode = c("driving", "transit", "walking", "bicycling"),
  arrival_time = NULL,
  departure_time = NULL,
  alternatives = FALSE,
  avoid = c(NA, "tolls", "highways", "ferries", "indoor"),
  region = NULL,
  traffic_model = c("best_guess", "pessimistic", "optimistic"),
  transit_mode = c("bus", "subway", "train", "tram"),
  transit_routing_preference = c(NA, "less_walking", "fewer_transfers"),
  language = NULL,
  key,
  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, default is FALSE)

avoid  NA (default, means avoid nothing) 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)

traffic_model  The traffic model, one of: "best_guess" (the default), "pessimistic", "optimistic". The traffic_model parameter is only taken into account when departure_time is specified!

transit_mode  Transit preferred mode, one or more of: "bus", "subway", "train" or "tram"

transit_routing_preference  Transit route preference. NA (default, means no preference) or one of: "less_walking" or "fewer_transfers"

language  The language in which to return directions. See https://developers.google.com/maps/faq#languagesupport for list of language codes.

key  Google APIs key

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/overview

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,
  key = key
)

# Comparing traffic models
doc = mp_directions(
  origin = "Beer-Sheva",
  destination = "Tel Aviv",
  departure_time = Sys.time() + as.difftime(1, units = "hours"),
  traffic_model = "best_guess",
  key = key
)
mp_get_routes(doc)$duration_in_traffic_text
doc = mp_directions(
  origin = "Beer-Sheva",
  destination = "Tel Aviv",
  departure_time = Sys.time() + as.difftime(1, units = "hours"),
  traffic_model = "optimistic",
  key = key
)
mp_get_routes(doc)$duration_in_traffic_text
doc = mp_directions(
  origin = "Beer-Sheva",
  destination = "Tel Aviv",
  departure_time = Sys.time() + as.difftime(1, units = "hours"),
  traffic_model = "best_guess",
  key = key
)
mp_geocode

```r
destination = "Tel Aviv",
departure_time = Sys.time() + as.difftime(1, units = "hours"),
traffic_model = "pessimistic",
key = key
)
mp_get_routes(doc)$duration_in_traffic_text

## End(Not run)
```

---

**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,  
  quiet = FALSE,  
  timeout = 10  
)
```

**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 non-missing
- **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)
- **quiet**  
  Logical; suppress printing geocode request statuses
- **timeout**  
  numeric of length 1, number of seconds to timeout, passed to `curls` connecttimeout option. Default is 10 seconds
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/overview

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
text_file = readLines("~/key")

# Basic use
addresses = c("Rehovot", "Beer-Sheva", "New-York")
doc = mp_geocode(addresses, key = key)
pnt = mp_get_points(doc)
print(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

mp_get_bounds(doc)

#### Arguments

doc XML document with Google Maps Geocode API response

#### Value

sf Polygonal layer representing bounds of geocoded locations. In cases when there is more than one response per address, only first response is considered.

#### Examples

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

## Not run:

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

# Get bounds
doc = mp_geocode(addresses = c("Tel-Aviv", "Rehovot", "Beer-Sheva"), region = "il", key = key)
b = mp_get_bounds(doc)
```
### 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 = c("distance_m", "distance_text", "duration_s", "duration_text",
             "duration_in_traffic_s", "duration_in_traffic_text")
)
```

**Arguments**

- `doc` XML document with Google Maps Distance Matrix API response
- `value` Value to extract, one of: "distance_m" (the 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 API 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 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:
# Text file with API key
```
key = readLines("~/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")

## End(Not run)

---

**mp_get_points**

Extract geocoded points from Google Maps Geocode API response

**Description**

Extract geocoded points from Google Maps Geocode API response

**Usage**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>XML document with Google Maps Geocode API response</td>
</tr>
<tr>
<td>all_results</td>
<td>The geocoder may return several results when address queries are ambiguous. Should all results be returned (TRUE), or just the first one (FALSE, default)?</td>
</tr>
</tbody>
</table>
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)
## Not run:
key = readLines("~/key")
doc = mp_geocode(addresses = c("Rehovot", "Beer-Sheva", "New-York"), key = key)
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)
```

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),
  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(
  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_map

*Get static map from the Maps Static API*

#### Description

Download a static map from the Maps Static API, given map center and zoom level.

#### Usage

```r
mp_map(
  center,  # Center of the map
  zoom = 10L,  # Zoom level
  maptype = c("roadmap", "satellite", "terrain", "hybrid"),
  size = c(640L, 640L),
  scale = 2L,
  style = NULL,
  key,
  quiet = FALSE
)
```

#### Arguments

- `center`: Character of length 1 of the form "lat,lon" or a geometry of class `sfg`, `sfc` or `sf`. If `center` is a geometry, the center of the geometry bounding box is passed as map center. Missing Coordinate Reference System (CRS) is assumed WGS84.
- `zoom`: Zoom level, a positive integer or zero. The appropriate range is 0 to 21. Defaults to '10'.
- `maptype`: Map type, one of: "roadmap", "satellite", "terrain", "hybrid".
- `size`: Numeric of length 2, the width and height of the map in pixels. The default is the maximum size allowed (640x640). The final dimensions of the image are affected by `scale`.
- `scale`: Integer, factor to multiply 'size' and determine the final image size. Allowed values are 1 and 2, defaults to 2.
- `style`: List of named character vector(s) specifying style directives. The full style reference is available at https://developers.google.com/maps/documentation/maps-static/style-reference, see examples below.
- `key`: Google APIs key
- `quiet`: Logical; suppress printing URL for Google Maps API call (e.g. to hide API key)

#### Value

A `stars` raster with the requested map, in Web Mercator CRS (EPSG:3857).
Examples

## Not run:

```r
library(stars)
key = readLines("~/key")

# Using coordinates
r = mp_map("31.253205,34.791914", 14, key = key)
plot(r)

# Using 'sfc' point - WGS84
pnt = st_point(c(34.791914, 31.253205))
pnt = st_sfc(pnt, crs = 4326)
r = mp_map(pnt, 14, key = key)
plot(r)

# Using 'sfc' point - UTM
pnt = st_point(c(34.791914, 31.253205))
pnt = st_sfc(pnt, crs = 4326)
pnt = st_transform(pnt, 32636)
r = mp_map(pnt, 14, key = key)
plot(r)

# Using 'sfc' polygon
pnt = st_point(c(34.791914, 31.253205))
pnt = st_sfc(pnt, crs = 4326)
pol = st_buffer(pnt, 0.01)
r = mp_map(pol, 14, key = key)
plot(r)

# 'ggplot2'
library(ggplot2)
cols = attr(r[[1]], "colors")
ggplot() +
  geom_stars(data = r, aes(x = x, y = y, fill = color)) +
  scale_fill_manual(values = cols, guide = FALSE) +
  coord_sf()
```

## 'ggplot2' - map types

```r
r1 = mp_map(pnt, 14, maptype = "roadmap", key = key)
r2 = mp_map(pnt, 14, maptype = "satellite", key = key)
r3 = mp_map(pnt, 14, maptype = "terrain", key = key)
r4 = mp_map(pnt, 14, maptype = "hybrid", key = key)
cols1 = attr(r1[[1]], "colors")
cols2 = attr(r2[[1]], "colors")
cols3 = attr(r3[[1]], "colors")
cols4 = attr(r4[[1]], "colors")
```
```r
theme1 = theme(
  axis.text = element_blank(),
  axis.title = element_blank(),
  axis.ticks = element_blank()
)
g1 = ggplot() +
  geom_stars(data = r1, aes(x = x, y = y, fill = color)) +
  scale_fill_manual(values = cols1, guide = FALSE) +
  coord_sf() +
  ggtitle("roadmap") +
  theme1
g2 = ggplot() +
  geom_stars(data = r2, aes(x = x, y = y, fill = color)) +
  scale_fill_manual(values = cols2, guide = FALSE) +
  coord_sf() +
  ggtitle("satellite") +
  theme1
g3 = ggplot() +
  geom_stars(data = r3, aes(x = x, y = y, fill = color)) +
  scale_fill_manual(values = cols3, guide = FALSE) +
  coord_sf() +
  ggtitle("terrain") +
  theme1
g4 = ggplot() +
  geom_stars(data = r4, aes(x = x, y = y, fill = color)) +
  scale_fill_manual(values = cols4, guide = FALSE) +
  coord_sf() +
  ggtitle("hybrid") +
  theme1
g1 + g2 + g3 + g4

# styled maps
nl = list(
  c(feature = 'all', element = 'labels', visibility = 'off')
)
nb = list(
  c(feature = 'poi.business', visibility = 'off'),
  c(feature = 'poi.medical', visibility = 'off')
)
r_nl = mp_map(pnt, 14, key = key, style = nl)
plot(r_nl)
r_nb = mp_map(pnt, 14, key = key, style = nb)
plot(r_nb)
```

---

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

**Description**

Get distance matrix from the Google Maps Distance Matrix API
Usage

mp_matrix(
  origins,
  destinations,
  mode = c("driving", "transit", "walking", "bicycling"),
  arrival_time = NULL,
  departure_time = NULL,
  avoid = c(NA, "tolls", "highways", "ferries", "indoor"),
  region = NULL,
  traffic_model = c("best_guess", "pessimistic", "optimistic"),
  transit_mode = c("bus", "subway", "train", "tram"),
  key,
  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

avoid NA (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)

traffic_model  The traffic model, one of: "best_guess" (the default), "pessimistic", "optimistic". The traffic_model parameter is only taken into account when departure_time is specified!

transit_mode  Transit preferred mode, one or more of: "bus", "subway", "train" or "tram"

key  Google APIs key

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/overview

Examples

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

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

# Using 'data.frame' input
doc = mp_matrix(
  origins = rbind(c(34.811, 31.892), c(35.212, 31.769)),
  destinations = c(34.781, 32.085),
  key = key
)

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

# Setting transit modes
locations = c("Tel-Aviv", "Beer-Sheva", "Eilat")
doc = mp_matrix(
  origins = locations,
  destinations = locations,
  key = key,
  mode = "transit",
  transit_mode = "train"
)

## End(Not run)
Usage

```r
## S3 method for class 'mapsapi_map'
plot(x, ...)
```

Arguments

- `x`: Map object of class `stars` and `mapsapi_map` obtained from function `mp_map`
- `...`: Further arguments passed to `plot.stars`

Description

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

Usage

```r
response_directions_driving
```

Format

A list obtained using `as_list` on XML response

Note

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

Examples

```r
library(xml2)
doc = as_xml_document(response_directions_driving)
```
**response_directions_transit**

*Sample response from Google Maps Directions API*

**Description**

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

**Usage**

```r
response_directions_transit
```

**Format**

A list obtained using `as_list` on XML response

**Note**

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

**Examples**

```r
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**

```r
response_geocode
```

**Format**

A list obtained using `as_list` on XML response

**Examples**

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

Sample response from Maps Static API (as ‘stars’ raster)

Description
A stars raster with a static image of Beer-Sheva from the Maps Static API

Usage
response_map

Format
A stars raster with two dimensions x and y and a color table

Examples
library(stars)
plot(response_map)

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 list obtained using as_list on XML response

Examples
library(xml2)
doc = as_xml_document(response_matrix)
Index

* datasets
  response_directions_driving, 18
  response_directions_transit, 19
  response_geocode, 19
  response_map, 20
  response_matrix, 20

  mp_directions, 2
  mp_geocode, 5
  mp_get_bounds, 6, 7
  mp_get_matrix, 8, 16
  mp_get_points, 6, 9
  mp_get_routes, 3, 10
  mp_get_segments, 3, 11
  mp_map, 13, 17, 18
  mp_matrix, 8, 15

  plot.mapsapi_map, 17

  response_directions_driving, 18, 19
  response_directions_transit, 18, 19
  response_geocode, 19
  response_map, 20
  response_matrix, 20