Package 'gmapsdistance'

October 22, 2026

Type Package

Title Distance and Travel Time Between Two Points from Google Maps

Version 4.1.0

Date 2026-10-22

URL https://github.com/jlacko/gmapsdistance

BugReports https://github.com/jlacko/gmapsdistance/issues

Description Get distance and travel time between two points from Google Maps. Supports both the legacy 'Distance Matrix API' (XML endpoint) and the newer 'Routes API' (REST).

License GPL (>= 3)

Depends R (>= 4.1.0)

Imports RCurl, XML, httr, jsonlite, methods

Encoding UTF-8

Suggests covr, knitr, rmarkdown, vctrs, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Rodrigo Azuero Melo [aut],
David Zarruk [aut],
Jindra Lacko [cre] (ORCID: <https://orcid.org/0000-0002-0375-5156>)

Maintainer Jindra Lacko <jindra.lacko@gmail.com>

Repository CRAN

Date/Publication 2026-10-22 16:40:09 UTC

Contents

distance_routes 2
get.api.key 5
gmapsdistance 5
set.api.key 8

Index 9

`distance_routes` *Compute Pairwise Distances with Google Maps*

Description

The function distance_routes uses the Compute Route Matrix method of the Google Maps Routes API to compute the distance and travel time between pairs of points: the first origin to the first destination, the second origin to the second destination and so on. In order to be able to use the function you will need an API key and enable the Routes API in the Google Developers Console. For more information about how to get a key, go to https://developers.google.com/maps/documentation/routes/get-api-key For more information about the Google Maps Routes API go to https://developers.google.com/maps/documentation/r

Usage

```
distance_routes(
  origin,
  destination,
  mode = "DRIVE",
  key = get.api.key(),
  avoid = NULL,
  traffic_model = NULL,
  departure = NULL,
  arrival = NULL
)
```

Arguments

`origin`	A string or vector of strings containing the description of the starting point(s), such as an address or a place name. Coordinates are also a valid input, as latitude and longitude in decimal degrees separated by a comma, e.g. "50.087,14.421" (spaces around the comma are fine). Note that the comma is required: unlike `gmapsdistance()`, coordinates separated by a space only, such as "50.087 14.421", are not recognized and Google rejects the request with an error.
`destination`	A string or vector of strings containing the description of the end point(s). Should be the same format as the variable "origin", including the comma between latitude and longitude of coordinates. Origins and destinations are paired, so the two vectors must be of the same length; a difference in lengths is an error.
`mode`	A string containing the mode of transportation desired, using the travel mode names of the Routes API. One of "DRIVE" (default), "WALK", "BICYCLE", "TRANSIT", "TWO_WHEELER" or "TRUCK". Note that "TWO_WHEELER" (motorized two wheelers) and "TRUCK" are supported only in some regions; elsewhere the status is "ZERO_RESULTS".

Note also that these values differ from the modes of transportation of `gmapsdistance()` ("driving", "walking", "bicycling", "transit"), due to the move from the legacy Distance Matrix API to the Routes API. The legacy values are rejected with an error pointing to their Routes API equivalent.

`key` Google Maps API key. Defaults to the key set by `set.api.key("THISISMYKEY")`.

`avoid` A string or vector of strings with route features to avoid, applied to all pairs. Any of "tolls", "highways" and "ferries" (for mode "DRIVE", "TWO_WHEELER" or "TRUCK") or "indoor" (for mode "WALK"); they correspond to the `avoidTolls`, `avoidHighways`, `avoidFerries` and `avoidIndoor` route modifiers of the Routes API. Default is `NULL`, i.e. nothing is avoided.

Avoiding is a preference, not a guarantee: Google avoids the features where reasonable, so a route may still include them.

`traffic_model` The assumptions to use when estimating travel time in traffic, using the traffic model names of the Routes API. One of "BEST_GUESS", "PESSIMISTIC" or "OPTIMISTIC"; defined only for mode "DRIVE". Default is `NULL`, i.e. traffic is not considered.

Traffic is estimated for the departure time, or for the time of the request when no departure is given. Note that requests with a traffic model use traffic aware routing, which Google bills at a higher rate, and that the route itself (and so the distance) may differ from the route without traffic.

`departure` The departure time, as a date-time (`POSIXct`) - a single time for all pairs, or one per origin / destination pair. Default is `NULL`, i.e. departing at the time of the request.

The time has to be in the future, except for mode "TRANSIT". For modes "DRIVE", "TWO_WHEELER" and "TRUCK" a departure time turns on traffic aware routing, which Google bills at a higher rate; `duration_traffic` is then the travel time in traffic at the time of departure.

Note that `as.POSIXct()` without a `tz` argument uses the time zone of your computer; give the time zone of the trip explicitly, e.g. `as.POSIXct("2026-11-02 08:00", tz = "Europe/London")`.

`arrival` The arrival time, as a date-time (`POSIXct`) - a single time for all pairs, or one per origin / destination pair; defined only for mode "TRANSIT", and may be in the past. Default is `NULL`, i.e. no arrival time. Departure and arrival times can not both be given.

Details

Request level problems (such as an invalid API key or an exhausted quota) raise an error carrying the message returned by Google. Route level problems (an origin or destination that can not be geocoded, or no route between the two) are reported per pair in the `status` element, with distance and duration set to `NA`. If the API can not be reached at all the function fails gracefully: it emits a message and returns `NA`.

Value

A data frame with one row per origin / destination pair, in the order of the input, and 8 columns; or `NA` if the API could not be reached:

- **origin**: the origins, as provided on input.
- **destination**: the destinations, as provided on input.
- **mode**: the mode of transportation, as provided on input.
- **status**: route status; "OK", "NOT_FOUND" (origin or destination could not be geocoded) or "ZERO_RESULTS" (no route between the points). Note that these differ from the status values of `gmapsdistance()`.
- **distance_text**: human readable distance, e.g. "4.2 km".
- **distance_value**: distance in meters; `NA` when status is not "OK".
- **duration_base**: travel time in seconds, without considering traffic; `NA` when status is not "OK".
- **duration_traffic**: travel time in seconds, considering traffic as estimated by the traffic model; the same as `duration_base` when no traffic model is given. `NA` when status is not "OK".

See Also

`gmapsdistance` for all combinations of origins and destinations.

Examples

```
## Not run:
# walking distance between two London stations
distance_routes(origin = "King's Cross St. Pancras, London, UK",
                destination = "Victoria, London, UK",
                mode = "WALK")

# driving distances for two pairs: London to Paris and Prague to Vienna
distance_routes(origin = c("London, UK", "Prague, CZ"),
                destination = c("Paris, FR", "Vienna, AT"))

# driving distance from Washington DC to NYC, avoiding toll roads
distance_routes(origin = "Washington DC",
                destination = "New York City NY",
                avoid = "tolls")

# driving time from King's Cross to Heathrow, expecting bad traffic
distance_routes(origin = "King's Cross St. Pancras, London, UK",
                destination = "Heathrow Airport, London, UK",
                traffic_model = "PESSIMISTIC")

# the same, departing in the morning rush hour, London time
distance_routes(origin = "King's Cross St. Pancras, London, UK",
                destination = "Heathrow Airport, London, UK",
                departure = as.POSIXct("2026-11-02 08:00", tz = "Europe/London"))

# public transport, arriving at 9 AM
distance_routes(origin = "King's Cross St. Pancras, London, UK",
                destination = "Heathrow Airport, London, UK",
                mode = "TRANSIT",
                arrival = as.POSIXct("2026-11-02 09:00", tz = "Europe/London"))
```

```
## End(Not run)
```

`get.api.key` *Get the Google Maps API key*

Description

This function returns the user's Google Maps API key that was defined with `set.api.key`.

Usage

```
get.api.key()
```

Value

the user's api key

`gmapsdistance` *gmapsdistance*

Description

Compute Distance with Google Maps

Usage

```
gmapsdistance(
    origin,
    destination,
    combinations = "all",
    mode = "driving",
    key = get.api.key(),
    shape = "wide",
    avoid = "",
    departure = "",
    dep_date = "",
    dep_time = "",
    traffic_model = "None",
    arrival = "",
    arr_date = "",
    arr_time = ""
)
```

Arguments

origin	A string or vector of strings containing the description of the starting point(s). Should be inside of quotes (""). Coordinates in LAT-LONG format are also a valid input as long as they can be identified by Google Maps.
destination	A string or vector of strings containing the description of the end point(s). Should be the same format as the variable "origin".
combinations	When the origin and destination entries are vectors, the user can specify if the function computes all possible combinations between origins and destinations, or only pairwise distance and times. Should be inside of double quotes (",") and one of the following: "all", "pairwise". If the combinations is set to "pairwise", the origin and destination vectors must have the same length.
mode	A string containing the mode of transportation desired. Should be inside of double quotes (",") and one of the following: "bicycling", "walking", "transit" or "driving".
key	In order to use the Google Maps Distance Matrix API it is necessary to have an API key. The key should be inside of quotes. Example: "THISISMYKEY". This key an also be set using `set.api.key("THISISMYKEY")`.
shape	A string that specifies the shape of the distance and time matrices to be returned. Should be inside of double quotes (",") and one of the following: "long" or "wide". If the function is used to find the distance/time for one origin and one destination, the shape does not matter. If there is more than one city as origin or destination, "long" will return a matrix in long format and "wide" will return a matrix in wide format. The shape is set as wide be default.
avoid	When the mode is set to "driving", the user can find the time and distance of the route by avoiding tolls, highways, indoor and ferries. Should be inside of double quotes (",") and one of the following: "tolls", "highways", "ferries", "indoor".
departure	The time and distance can be computed at the desired time of departure. The option departure is the number of seconds since January 1, 1970 00:00:00 UTC. Alternatively, the user can use the dep_date and dep_time options to set the departure date and time. If no value is set for departure, dep_date and dep_time, the departure time is set to the present. Note that API calls that satisfy both of these conditions: 1) departure time is specified and 2) travel mode equals "driving" (either directly or via fallback to default) incur higher cost.
dep_date	Instead of using the departure option, the user can set the departure date and time using dep_date and dep_time options. If no value is set for departure, dep_date and dep_time, the departure time is set to the present.
dep_time	Instead of using the departure option, the user can set the departure date and time using dep_date and dep_time options.

If no value is set for departure, dep_date and dep_time, the departure time is set to the present.

`traffic_model` When the mode is set to "driving" and a departure time is provided the user can find the times and distances using different traffic models. Should be a string and one of the following: "optimistic", "pessimistic", "best_guess" or "None".

Default is "None". The traffic model is not defined for other modes of transport than driving, and providing it for e.g. mode = "walking" is illegal.

`arrival` For transportation mode "transit" the time and distance can be computed to arrive at a predetermined time. The option arrival is the number of seconds since January 1, 1970 00:00:00 UTC. Alternatively, the user can use the arr_date and arr_time options to set the arrival date and time.

For transport modes other than "transit" the use of arrival (and arr_date + arr_time) is illegal.

The user cannot input both departure and arrival times.

`arr_date` Instead of using the arrival option, the user can set the arrival date and time using arr_date and arr_time options.

The user cannot input both departure and arrival times.

`arr_time` Instead of using the arrival option, the user can set the arrival date and time using arr_date and arr_time options.

The user cannot input both departure and arrival times.

Details

The function gmapsdistance uses the Google Maps Distance Matrix API in order to compute the distance(s) and time(s) between two points. In order to be able to use the function you will need an API key and enable the Distance Matrix API in the Google Developers Console For more information about how to get a key, go to https://developers.google.com/maps/documentation/distance-matrix/get-api-key#key For more information about the Google Maps Distance Matrix API go to https://developers.google.com/maps/documentation/distance-matrix/intro?hl=en

Value

A list with 3 named elements:

- **Distance**: distance between the points provided, taking into account mode of transport.
- **Time**: time of travel, taking into account mode of transport.
  In case departure time was specified, and mode was set to "driving" (either directly or via fallback default) the travel time includes delays due to traffic.
- **Status**: status of the API call.

Examples

```
## Not run:
# distance from Washington DC to NYC
gmapsdistance(origin = "Washington DC",
              destination = "New York City NY")
```

```
# distance matrix between 3 US cities
gmapsdistance(origin = c("Washington DC", "New York NY", "Seattle WA"),
              destination = c("Washington DC", "New York NY", "Seattle WA"))$Distance

## End(Not run)
```

`set.api.key` *Set the Google Maps API key*

Description

This function stores a user's Google Maps API key as the package's environmental variable

Usage

```
set.api.key(key)
```

Arguments

`key`	is the user's Google Maps API key

Examples

```
#DONTRUN
set.api.key("MY-GOOGLE-MAPS-API-KEY")
```

Index

distance_routes, 2

get.api.key, 5
gmapsdistance, *4*, 5

set.api.key, 8