Package ‘mapboxapi’

April 12, 2022

Type Package
Title R Interface to ‘Mapbox’ Web Services
Date 2022-04-12
Version 0.3.3
Maintainer Kyle Walker <kyle@walker-data.com>
Description Includes support for ‘Mapbox’ Navigation APIs, including directions, isochrones, and route optimization; the Search API for forward and reverse geocoding; the Maps API for interacting with ‘Mapbox’ vector tilesets and visualizing ‘Mapbox’ maps in R; and the ‘tippecanoe’ tile-generation utility.
See <https://docs.mapbox.com/api/> for more information about the ‘Mapbox’ APIs.
License MIT + file LICENSE
Encoding UTF-8
Depends R (>= 3.3.0)
Imports httr, sf, jsonlite, purrr, curl, dplyr (>= 1.0.0), tidyr (>= 1.0.0), aws.s3, stringi, slippymath, protolite, rlang, geojsonsf, magick, leaflet, units, raster, png, jpeg
Suggests ggspatial, mapdeck, tigris, tidycensus, tmapping
RoxygenNote 7.1.2
NeedsCompilation no
Author Kyle Walker [aut, cre], 
Eli Pousson [ctb]
Repository CRAN
Date/Publication 2022-04-12 21:32:28 UTC

R topics documented:

  addMapboxTiles ......................................................... 2
  check_upload_status .............................................. 3
  get_static_tiles .................................................... 4
  get_style ............................................................. 6
  get_vector_tiles ..................................................... 6
addMapboxTiles

Use a Mapbox style in a Leaflet map

Description

Use a Mapbox style in a Leaflet map

Usage

addMapboxTiles(
  map,
  style_id,
  username,
  style_url = NULL,
  scaling_factor = c("1x", "0.5x", "2x"),
  access_token = NULL,
  layerId = NULL,
  group = NULL,
  options = leaflet::tileOptions(),
  data = leaflet::getMapData(map),
  attribution = TRUE
)

Arguments

map A map widget object created by leaflet::leaflet()
style_id The style ID of a Mapbox style
username A Mapbox username
style_url A Mapbox style URL
check_upload_status

scaling_factor  The scaling factor to use when rendering the tiles. A scaling factor of 1 (the default) returns 512px by 512px tiles. A factor of 0.5 returns 256x256 tiles, and a factor of 2 returns 1024x1024 tiles.

access_token  Your Mapbox access token; can be set with mb_access_token().

layerId  the layer ID

group  The name of the group the Mapbox tile layer should belong to (for use in Shiny and to modify layers control in a Leaflet workflow)

options  A list of extra options (optional)

data  The data object used to derive argument values; can be provided to the initial call to leaflet::leaflet()

attribution  If ‘TRUE’, pass a standard attribution to [leaflet::addTiles]. If ‘FALSE’, attribution is ‘NULL’. attribution can also be a character string including HTML.

Value

A pointer to the Mapbox Static Tiles API which will be translated appropriately by the leaflet R package.

Examples

## Not run:

library(leaflet)
library(mapboxapi)

leaflet() %>%
  addMapboxTiles(
    style_id = "light-v9",
    username = "mapbox"
  ) %>%
  setView(
    lng = -74.0051,
    lat = 40.7251,
    zoom = 13
  )

## End(Not run)

---

**check_upload_status**  Check the status of a Mapbox upload

**Description**

Check the status of a Mapbox upload
Usage

check_upload_status(upload_id, username, access_token = NULL)

Arguments

upload_id  The upload ID
username    Your account’s username
access_token Your Mapbox access token

get_static_tiles Get static tiles from a Mapbox style for use as a basemap

Description

This function queries the Mapbox Static Tiles API and composites the tiles as a raster suitable for use as a basemap in tmap or ggplot2 (with the layer_spatial() function in ggspatial). It returns a raster layer that corresponds either to an input bounding box or a buffered area around an input shape.

Usage

get_static_tiles(
  location,
  zoom,
  style_id,
  username,
  style_url = NULL,
  scaling_factor = c("1x", "2x"),
  buffer_dist = 5000,
  crop = TRUE,
  access_token = NULL
)

Arguments

location An input location for which you would like to request tiles. Can be a length-4 vector representing a bounding box, or an sf object. If an input sf object is supplied, use the buffer_dist argument to control how much area you want to capture around the layer. While the input sf object can be in an arbitrary coordinate reference system, if a length-4 bounding box vector is supplied instead it must represent WGS84 longitude/latitude coordinates and be in the order c(xmin, ymin, xmax, ymax).
zoom The zoom level for which you’d like to return tiles.
style_id A Mapbox style ID; retrieve yours from your Mapbox account.
username A Mapbox username.
### get_static_tiles

- **style_url**: A Mapbox style URL.
- **scaling_factor**: The scaling factor to use; one of "1x" or "2x".
- **buffer_dist**: The distance to buffer around an input sf object for determining tile extent, specified in meters. Defaults to 5000.
- **crop**: Whether or not to crop the result to the specified bounding box or buffer area. Defaults to `TRUE`; `FALSE` will return the extent of the overlapping tiles.
- **access_token**: Your Mapbox access token. Supply yours here or set globally with the `mb_access_token()` function.

### Value

A raster layer of tiles from the requested Mapbox style representing the area around the input location. The raster layer is projected in the Web Mercator coordinate reference system.

### Examples

#### Not run:

```r
library(mapboxapi)
library(tigris)
library(tmap)
library(ggspatial)
library(ggplot2)

ny_tracts <- tracts("NY", "New York", cb = TRUE)

ny_tiles <- get_static_tiles(
  location = ny_tracts,
  zoom = 10,
  style_id = "light-v9",
  username = "mapbox"
)

# tmap usage:
tm_shape(ny_tiles) +
  tm_rgb() +
  tm_shape(ny_tracts) +
  tm_polygons(alpha = 0.5, col = "navy") +
  tm_credits("Basemap (c) Mapbox, (c) OpenStreetMap",
             position = c("RIGHT", "BOTTOM")
)

# ggplot2 usage:
ggplot() +
  layer_spatial(ny_tiles) +
  geom_sf(data = ny_tracts, fill = "navy", alpha = 0.5) +
  theme_void() +
  labs(caption = "Basemap (c) Mapbox, (c) OpenStreetMap")
```

#### End(Not run)
---

**get_style**  
*Get information about a style from your Mapbox account*

**Description**

Get information about a style from your Mapbox account

**Usage**

```r
get_style(style_id, username, style_url = NULL, access_token = NULL)
```

**Arguments**

- **style_id**: A style ID
- **username**: A Mapbox username
- **style_url**: Mapbox style URL
- **access_token**: A Mapbox public or secret access token; set with `mb_access_token()`

**Value**

A list of information about your selected style.

---

**get_vector_tiles**  
*Retrieve vector tiles from a given Mapbox tileset*

**Description**

Retrieve vector tiles from a given Mapbox tileset

**Usage**

```r
get_vector_tiles(tileset_id, location, zoom, access_token = NULL)
```

**Arguments**

- **tileset_id**: The name of the tileset ID; names can be retrieved from your Mapbox account
- **location**: The location for which you’d like to retrieve tiles. If the input is an sf object, the function will return data for all tiles that intersect the object’s bounding box. If the input is a coordinate pair or an address, data will be returned for the specific tile that contains the input.
- **zoom**: The zoom level of the request; larger zoom levels will return more detail but will take longer to process.
- **access_token**: A Mapbox access token; can be set with `mb_access_token()`.
Value

A list of sf objects representing the different layer types found in the requested vector tiles.

Examples

```r
## Not run:

library(mapboxapi)
library(ggplot2)

vector_extract <- get_vector_tiles(
  tileset_id = "mapbox.mapbox-streets-v8",
  location = c(-73.99405, 40.72033),
  zoom = 15
)

ggplot(vector_extract$building$polygons) +
  geom_sf() +
  theme_void()

## End(Not run)
```

---

### layer_static_mapbox

**Make a ggplot2 layer with static_mapbox and ggspatial**

### Description

Make a ggplot2 layer with static_mapbox and ggspatial

### Usage

```r
layer_static_mapbox(
  location = NULL,
  buffer_dist = 1000,
  style_id,
  username,
  style_url = NULL,
  overlay_sf = NULL,
  overlay_style = NULL,
  overlay_markers = NULL,
  width = NULL,
  height = NULL,
  scale = 0.5,
  scaling_factor = c("1x", "2x"),
  attribution = TRUE,
  logo = TRUE,
  before_layer = NULL,
)```
access_token = NULL,
...
)

Arguments
location An input location for which you would like to request tiles. Can be a length-4 vector representing a bounding box, or an sf object. If an input sf object is supplied, use the buffer_dist argument to control how much area you want to capture around the layer. While the input sf object can be in an arbitrary coordinate reference system, if a length-4 bounding box vector is supplied instead it must represent WGS84 longitude/latitude coordinates and be in the order c(xmin, ymin, xmax, ymax).

buffer_dist The distance to buffer around an input sf object for determining static map, specified in meters. Defaults to 1000.

style_id A style ID (required if style_url is NULL).

username A Mapbox username (required if style_url is NULL).

style_url A Mapbox style url; defaults to NULL.

overlay_sf The overlay sf object (optional). The function will convert the sf object to GeoJSON then plot over the basemap style. Spatial data that are too large will trigger an error, and should be added to the style in Mapbox Studio instead.

overlay_style A named list of vectors specifying how to style the sf overlay. Possible names are "stroke", "stroke-width", "stroke-opacity", "fill", and "fill-opacity". The fill and stroke color values should be specified as six-digit hex codes, and the opacity and width values should be supplied as floating-point numbers.

overlay_markers The prepared overlay markers (optional). See the function prep_overlay_markers for more information on how to specify a marker overlay.

width The map width; defaults to NULL

height The map height; defaults to NULL

scale ratio to scale the output image; 'scale = 1' will return the largest possible image. defaults to 0.5

scaling_factor The scaling factor of the tiles; either "1x" (the default) or "2x"

attribution Controls whether there is attribution on the image. Defaults to 'TRUE'. Note: If attribution=false, the watermarked attribution is removed from the image. You still have a legal responsibility to attribute maps that use OpenStreetMap data, which includes most maps from Mapbox. If you specify attribution=false, you are legally required to include proper attribution elsewhere on the webpage or document.

logo Controls whether there is a Mapbox logo on the image. Defaults to 'TRUE'.

before_layer A character string that specifies where in the hierarchy of layer elements the overlay should be inserted. The overlay will be placed just above the specified layer in the given Mapbox styles.

access_token A Mapbox access token; can be set with mb_access_token().

... additional parameters passed to layer_spatial
**list_styles**

*List styles in your Mapbox account*

**Description**

List styles in your Mapbox account

**Usage**

```r
list_styles(username, access_token = NULL)
```

**Arguments**

- `username` Your Mapbox username
- `access_token` Your Mapbox public or secret access token; set with `mb_access_token()`

**Value**

A data frame of information about styles in your Mapbox account

---

**mapboxapi**

*An R interface to Mapbox web services*

**Description**

Use Mapbox web services APIs for spatial data science and visualization projects in R. Usage of the package is governed by the Mapbox Terms of Service.

**Author(s)**

Kyle Walker
Install or retrieve a Mapbox access token in your .Renviron for repeated use

List tokens from a Mapbox account

Usage

mb_access_token(token, overwrite = FALSE, install = FALSE)

get_mb_access_token(
  token = NULL,
  default = c("MAPBOX_PUBLIC_TOKEN", "MAPBOX_SECRET_TOKEN"),
  secret_required = FALSE
)

list_tokens(
  username,
  default = NULL,
  limit = NULL,
  sortby = "created",
  usage = NULL,
  access_token = NULL
)

Arguments

token A Mapbox access token; can be public (starting with 'pk') or secret (starting with 'sk') scope, which the function will interpret for you.

overwrite Whether or not to overwrite an existing Mapbox access token. Defaults to FALSE.

install if TRUE, will install the key in your .Renviron file for use in future sessions. Defaults to FALSE.

default If TRUE, will only include the default token for an account. If FALSE, will include all other tokens except for the default. Defaults to NULL.

secret_required If TRUE, a secret token is required. If FALSE, the default token is provided first and the other token provided second if the first is unavailable.

username The Mapbox username for which you’d like to list access tokens.

limit The maximum number of tokens to return. Defaults to NULL.

sortby How to sort the returned tokens; one of "created" or "modified".
usage  If "pk", returns only public tokens; if "sk", returns only secret tokens. Defaults to NULL, which returns all tokens in the scope of the supplied access token.

access_token  A Mapbox access token. If left blank, will first check to see if you have a secret token stored in .Renviron, then a public token.

Value

A tibble of information about tokens in your Mapbox account.

Examples

```r
## Not run:
my_token <- "..." # The token generated from your Mapbox account
mb_access_token(my_token, install = TRUE)
Sys.getenv("MAPBOX_PUBLIC_TOKEN")

get_mb_access_token()

## End(Not run)

## Not run:
token_list <- list_tokens(
  username = "kwalkertcu", # You would use your own username here
  limit = 10,
  sortby = "modified" #'
)

## End(Not run)
```

mb_directions  Make a request to the Mapbox Directions API

Description

Make a request to the Mapbox Directions API

Usage

```r
mb_directions(
  input_data = NULL,
  origin = NULL,
  destination = NULL,
  profile = "driving",
  output = "sf",
  depart_at = NULL,
  alternatives = NULL,
  annotations = NULL,
  bearings = NULL,
)```
mb_directions

```r
continue_straight = NULL,
exclude = NULL,
geometries = NULL,
overview = "simplified",
radiuses = NULL,
approaches = NULL,
steps = NULL,
banner_instructions = NULL,
language = NULL,
roundabout_exits = NULL,
voice_instructions = NULL,
voice_units = NULL,
waypoint_names = NULL,
waypoint_targets = NULL,
waypoints = NULL,
walking_speed = NULL,
walkway_bias = NULL,
alley_bias = NULL,
access_token = NULL
```

**Arguments**

- **input_data**
  - An input dataset of class "sf", or a list of coordinate pairs for format c(longitude, latitude).
  - Cannot be used with an origin/destination pair.

- **origin**
  - An address or coordinate pair that represents the origin of your requested route.
  - Cannot be used with input_data.

- **destination**
  - An address or coordinate pair that represents the destination of your requested route.

- **profile**
  - One of "driving" (the default), "driving-traffic", "walking", or "cycling".

- **output**
  - One of "sf" (the default), which returns an sf LINESTRING representing the route geometry, or "full", which returns the full request from the Directions API as a list.

- **depart_at**
  - (optional) For the "driving" or "driving-traffic" profiles, the departure date and time to reflect historical traffic patterns. If "driving-traffic" is used, live traffic will be mixed in with historical traffic for dates/times near to the current time.
  - Should be specified as an ISO 8601 date/time, e.g. "2022-03-31T09:00".

- **alternatives**
  - Whether or not to return alternative routes with your request. If TRUE, a list of up to 3 possible routes will be returned.

- **annotations**
  - A comma-separated string of additional route metadata, which may include duration, distance, speed, and congestion. Must be used with overview = "full".

- **bearings**
  - A semicolon-delimited character string of bearings

- **continue_straight**
  - continue_straight

- **exclude**
  - Road types to exclude from your route; possible choices are 'toll', 'motorway', or 'ferry'. Defaults to NULL.
**geometries**  The route geometry format. If `output = 'sf'`, you will get back an sf object and you should leave this blank. If `output = 'full'`, the embedded route geometries will be polyline with five decimal place precision. 'polyline6' may also be specified.

**overview**  If left blank, defaults to 'simplified' for simplified geometry; the other option is 'full' which provides the most detailed geometry available.

**radiiuses**  A character string with semicolon-separated radii that specify the distance (in meters) to snap each input coordinate to the road network. Defaults to NULL.

**approaches**  A character string with semicolon-separated specifications for how to approach waypoints. Options include unrestricted and curb. Defaults to NULL which uses unrestricted for all waypoints.

**steps**  If TRUE, returns the route object split up into route legs with step-by-step instructions included. If FALSE or NULL (the default), a single line geometry representing the full route will be returned.

**banner_instructions**  Whether or not to return banner objects; only available when output = 'full' and steps = TRUE.

**language**  The language of the returned instructions (defaults to English). Available language codes are found at [https://docs.mapbox.com/api/navigation/#instructions-languages](https://docs.mapbox.com/api/navigation/#instructions-languages). Only available when steps = TRUE.

**roundabout_exits**  If TRUE, adds instructions for roundabout entrance and exit. Only available when steps = TRUE.

**voice_instructions**  Only available when steps = TRUE and output = 'full'.

**voice_units**  Only available when steps = TRUE and output = 'full'.

**waypoint_names**  Only available when steps = TRUE and output = 'full'.

**waypoint_targets**  Only available when steps = TRUE and output = 'full'.

**waypoints**  Only available when steps = TRUE and output = 'full'.

**walking_speed**  The walking speed in meters/second; available when profile = 'walking'.

**walkway_bias**  Can take values between -1 and 1, where negative numbers avoid walkways and positive numbers prefer walkways. Available when profile = 'walking'.

**alley_bias**  Can take values between -1 and 1, where negative numbers avoid alleys and positive numbers prefer alleys. Available when profile = 'walking'.

**access_token**  Your Mapbox access token; set with `mb_access_token()`

**Value**

An sf object (or list of sf objects), or full R list representing the API response.
Examples

```r
## Not run:
library(mapboxapi)
library(leaflet)

my_route <- mb_directions(
  origin = "10 Avenue de Wagram, 75008 Paris France",
  destination = "59 Rue de Tocqueville, 75017 Paris France",
  profile = "cycling",
  steps = TRUE,
  language = "fr"
)

leaflet(my_route) %>%
  addMapboxTiles(
    style_id = "light-v9",
    username = "mapbox"
  ) %>%
  addPolylines()
## End(Not run)
```

---

**mb_geocode**

*Geocode an address or place description using the Mapbox Geocoding API*

---

**Description**

Geocode an address or place description using the Mapbox Geocoding API

Perform reverse geocoding for a coordinate pair

**Usage**

```r
mb_geocode(
  search_text, 
  endpoint = "mapbox.places", 
  limit = 1, 
  types = NULL, 
  search_within = NULL, 
  language = NULL, 
  output = "coordinates", 
  access_token = NULL 
)
```

```r
mb_reverse_geocode(
  coordinates, 
  endpoint = "mapbox.places", 
  limit = 1, 
)```
mb_geocode

language = NULL,
types = NULL,
output = "text",
access_token = NULL

Arguments

search_text  The text to search, formatted as a character string. Can be an address, a location, or a description of a point of interest.

endpoint  One of 'mapbox.places' (the default) or mapbox.places-permanent. Per Mapbox’s terms of service, you are only allowed to save results and perform batch geocoding with the places-permanent endpoint.

limit  How many results to return; defaults to 1 (maximum 10).

types  A vector of feature types to limit to which the search should be limited. Available options include 'country', 'region', 'postcode', 'district', 'place', 'locality', 'neighborhood', 'address', and 'poi'. If left blank, all types will be searched.

search_within  An sf object, or vector representing a bounding box of format c(min_longitude, min_latitude, max_longitude, max_latitude) used to limit search results. Defaults to NULL.

language  The user's language, which can help with interpretation of queries. Available languages are found at https://docs.mapbox.com/api/search/#language-coverage.

output  one of "text" (the default), which will return a character string or list of character strings representing the returned results; output = "sf", returning an sf object; or "full", which will return a list with the full API response.

access_token  The Mapbox access token (required); can be set with mb_access_token.

coordinates  The coordinates of a location in format c(longitude, latitude) for which you’d like to return information.

Value

A character vector, list, or sf object representing the query results.

Examples

## Not run:

whitehouse <- mb_geocode("1600 Pennsylvania Ave, Washington DC")

## End(Not run)

## Not run:

mb_reverse_geocode(c(77.5958768, 12.9667046), limit = 5, types = "poi")

## End(Not run)
Generate isochrones using the Mapbox Navigation API

Description

This function returns isochrones from the Mapbox Navigation API, which are shapes that represent the reachable area around one or more locations within a given travel time. Isochrones can be computed for driving, walking, or cycling routing profiles, and can optionally be set to return distances rather than times. `mb_isochrone()` returns isochrones as simple features objects in the WGS 1984 geographic coordinate system.

Usage

```r
mb_isochrone(
  location,
  profile = "driving",
  time = c(5, 10, 15),
  distance = NULL,
  depart_at = NULL,
  access_token = NULL,
  denoise = 1,
  generalize = NULL,
  geometry = "polygon",
  output = "sf",
  rate_limit = 300,
  keep_color_cols = FALSE,
  id_column = NULL
)
```

Arguments

- **location**: A vector of form `c(longitude, latitude)`, an address that can be geocoded as a character string, or an `sf` object.
- **profile**: One of "driving", "walking", "cycling", or "driving-traffic". "driving" is the default.
- **time**: A vector of isochrone contours, specified in minutes. Defaults to `c(5, 10, 15)`. The maximum time supported is 60 minutes. Reflects traffic conditions for the date and time at which the function is called. If reproducibility of isochrones is required, supply an argument to the `depart_at` parameter.
- **distance**: A vector of distance contours specified in meters. If supplied, will supersede any call to the `time` parameter as time and distance cannot be used simultaneously. Defaults to `NULL`.
- **depart_at**: (optional) For the "driving" or "driving-traffic" profiles, the departure date and time to reflect historical traffic patterns. If "driving-traffic" is used, live traffic will be mixed in with historical traffic for dates/times near to the current time. Should be specified as an ISO 8601 date/time, e.g. "2022-03-31T09:00". If
NULL (the default), isochrones will reflect traffic conditions at the date and time when the function is called.

access_token  A valid Mapbox access token.
denoise  A floating-point value between 0 and 1 used to remove smaller contours. 1 is the default and returns only the largest contour for an input time.
generalize  A value expressed in meters of the tolerance for the Douglas-Peucker generalization algorithm used to simplify the isochrone shapes. If NULL (the default), the Mapbox API will choose an optimal value for you.
geometry  one of "polygon" (the default), which returns isochrones as polygons, or alternatively "linestring", which returns isochrones as linestrings.
output  one of "sf" (the default), which returns an sf object representing the isochrone(s), or "list", which returns the GeoJSON response from the API as an R list.
rate_limit  The rate limit for the API, expressed in maximum number of calls per minute. For most users this will be 300 though this parameter can be modified based on your Mapbox plan. Used when location is "sf".
keep_color_cols  Whether or not to retain the color columns that the Mapbox API generates by default (applies when the output is an sf object). Defaults to FALSE.
id_column  If the input dataset is an sf object, the column in your dataset you want to use as the isochrone ID. Otherwise, isochrone IDs will be identified by row index or position.

Value

An sf object representing the isochrone(s) around the location(s).

Examples

```r
## Not run:
library(mapboxapi)
library(mapdeck)
isochrones <- mb_isochrone("The Kremlin, Moscow Russia",
  time = c(4, 8, 12),
  profile = "walking"
)
mapdeck(style = mapdeck_style("light")) %>%
  add_polygon(
    data = isochrones,
    fill_colour = "time",
    fill_opacity = 0.5,
    legend = TRUE
  )
## End(Not run)
```
mb_matrix

Retrieve a matrix of travel times from the Mapbox Directions API

**Description**

Retrieve a matrix of travel times from the Mapbox Directions API

**Usage**

```r
mb_matrix(
  origins,
  destinations = NULL,
  profile = "driving",
  fallback_speed = NULL,
  access_token = NULL,
  duration_output = "minutes"
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>origins</td>
<td>The input coordinates of your request. Acceptable inputs include a list of coordinate pair vectors in c(x, y) format or an sf object. For sf linestrings or polygons, the distance between centroids will be taken.</td>
</tr>
<tr>
<td>destinations</td>
<td>The destination coordinates of your request. If NULL (the default), a many-to-many matrix using origins will be returned.</td>
</tr>
<tr>
<td>profile</td>
<td>One of &quot;driving&quot; (the default), &quot;driving-traffic&quot;, &quot;walking&quot;, or &quot;cycling&quot;.</td>
</tr>
<tr>
<td>fallback_speed</td>
<td>A value expressed in kilometers per hour used to estimate travel time when a route cannot be found between locations. The returned travel time will be based on the straight-line estimate of travel between the locations at the specified fallback speed.</td>
</tr>
<tr>
<td>access_token</td>
<td>A Mapbox access token (required)</td>
</tr>
<tr>
<td>duration_output</td>
<td>one of &quot;minutes&quot; (the default) or &quot;seconds&quot;</td>
</tr>
</tbody>
</table>

**Value**

An R matrix of source-destination travel times.

**Examples**

```r
## Not run:
library(mapboxapi)
library(tigris)
library(mapdeck)
```
mb_optimized_route

Return an optimized route for a series of input coordinates

Description

Return an optimized route for a series of input coordinates

Usage

mb_optimized_route(
  input_data,  # An input dataset of class "sf", or a list of coordinate pairs of format c(longitude, latitude). Must be between 2 and 12 coordinate pairs.
  profile = c("driving", "walking", "cycling", "driving-traffic"),
  output = "sf",
  source = c("any", "first"),
  destination = c("any", "last"),
  roundtrip = TRUE,
  annotations = NULL,
  approaches = NULL,
  bearings = NULL,
  distributions = NULL,
  language = NULL,
  overview = "simplified",
  radiuses = NULL,
  steps = NULL,
  access_token = NULL
)

Arguments

input_data: An input dataset of class "sf", or a list of coordinate pairs of format c(longitude, latitude). Must be between 2 and 12 coordinate pairs.
profile One of "driving" (the default), "driving-traffic", "walking", or "cycling".

output One of "sf" (the default), which returns an sf LINESTRING representing the route geometry, or "full", which returns the full request from the Directions API as a list.

source One of "any" (the default) or "first". If "any" is specified, any of the input coordinates may be used as the starting point. If "first" is specified, the first coordinate will be used.

destination One of "any" (the default) or "last". If "any" is specified, any of the input coordinates may be used as the ending point. If "last" is specified, the last coordinate will be used.

roundtrip If TRUE (the default), the route will start and end at the same point. roundtrip = FALSE only works when source is "first" and destination is "last". If FALSE is supplied here, the route will start at the first point in input_data and end at the last point.

annotations A comma-separated string of additional route metadata, which may include duration, distance, speed, and congestion. Must be used with overview = "full".

approaches A character string with semicolon-separated specifications for how to approach waypoints. Options include unrestricted and curb. Defaults to NULL which uses unrestricted for all waypoints.

bearings A semicolon-delimited character string of bearings.

distributions A semicolon-delimited character string of number pairs that specifies pick-up and drop-off locations. The first number indicates the index of the pick-up location, and the second number represents the index of the drop-off location.

language The language of the returned instructions (defaults to English). Available language codes are found at https://docs.mapbox.com/api/navigation/#instructions-languages. Only available when steps = TRUE.

overview If left blank, defaults to 'simplified' for simplified geometry; the other option is 'full' which provides the most detailed geometry available.

radiuses A character string with semicolon-separated radii that specify the distance (in meters) to snap each input coordinate to the road network. Defaults to NULL.

steps If TRUE, returns the route object split up into route legs with step-by-step instructions included. If FALSE or NULL (the default), a single line geometry representing the full route will be returned.

access_token Your Mapbox access token; set with mb_access_token()
library(sf)

to_visit <- data.frame(
  X = c(-0.209307, -0.185875, -0.216877, -0.233511, -0.234541),
  Y = c(5.556019, 5.58031, 5.582528, 5.566771, 5.550209)
)

st_as_sf(coords = c("X", "Y"), crs = 4326)

optimized_route <- mb_optimized_route(to_visit,
  profile = "driving-traffic"
)

## End(Not run)

---

**query_tiles**  
*Get information about features in a tileset using the Tilequery API*

**Description**

Get information about features in a tileset using the Tilequery API

**Usage**

```r
query_tiles(
  location,  
  tileset_id,  
  radius = 0,  
  limit = 5,  
  dedupe = TRUE,  
  geometry = NULL,  
  layers = NULL,  
  access_token = NULL
)
```

**Arguments**

- **location**  
The location for which you’d like to query tiles, expressed as either a length-2 vector of longitude and latitude or an address you’d like to geocode.

- **tileset_id**  
The tileset ID to query.

- **radius**  
The radius around the point (in meters) for which you’d like to query features. For point-in-polygon queries (e.g. "what county is my point located in?") the default of 0 should be used.

- **limit**  
How many features to return (defaults to 5). Can be an integer between 1 and 50.

- **dedupe**  
Whether or not to return duplicate features as identified by their IDs. The default, TRUE, will de-duplicate your dataset.
geometry     The feature geometry type to query - can be "point", "linestring", or "polygon". If left blank, all geometry types will be queried.
layers       A vector of layer IDs you’d like to query (recommended); if left blank will query all layers, with the limitation that at most 50 features can be returned.
access_token A Mapbox access token, which can be set with mb_access_token().

Value

An R list containing the API response, which includes information about the requested features. Parse the list to extract desired elements.

See Also

https://docs.mapbox.com/help/tutorials/find-elevations-with-tilequery-api/

Examples

```r
## Not run:
library(mapboxapi)
elevation <- query_tiles(
  location = "Breckenridge, Colorado",
  tileset_id = "mapbox.mapbox-terrain-v2",
  layer = "contour",
  limit = 50
)
max(elevation$features$properties$ele)
## End(Not run)
```

---

**static_mapbox**

Return a static Mapbox map from a specified style

Description

Return a static Mapbox map from a specified style

Prepare overlay markers for use in a Mapbox static map

Usage

static_mapbox(
  location = NULL,
  buffer_dist = 1000,
  style_id,
  username,
  **
```
style_url = NULL,
overlay_sf = NULL,
overlay_style = NULL,
overlay_markers = NULL,
longitude = NULL,
latitude = NULL,
zoom = NULL,
width = NULL,
height = NULL,
bearing = NULL,
pitch = NULL,
scale = 0.5,
scaling_factor = c("1x", "2x"),
 attribution = TRUE,
logo = TRUE,
before_layer = NULL,
access_token = NULL,
image = TRUE
}

prep_overlay_markers(
data = NULL,
 marker_type = c("pin-s", "pin-l", "url"),
 label = NA,
 color = NA,
 longitude = NULL,
 latitude = NULL,
 url = NA
)

Arguments

location  
An input location for which you would like to request tiles. Can be a length-4 vector representing a bounding box, or an sf object. If an input sf object is supplied, use the buffer_dist argument to control how much area you want to capture around the layer. While the input sf object can be in an arbitrary coordinate reference system, if a length-4 bounding box vector is supplied instead it must represent WGS84 longitude/latitude coordinates and be in the order c(xmin, ymin, xmax, ymax).

buffer_dist  
The distance to buffer around an input sf object for determining static map, specified in meters. Defaults to 1000.

style_id  
A style ID (required if style_url is NULL).

username  
A Mapbox username (required if style_url is NULL).

style_url  
A Mapbox style url; defaults to NULL.

overlay_sf  
The overlay sf object (optional). The function will convert the sf object to GeoJSON then plot over the basemap style. Spatial data that are too large will trigger an error, and should be added to the style in Mapbox Studio instead.
overlay_style A named list of vectors specifying how to style the sf overlay. Possible names are "stroke", "stroke-width", "stroke-opacity", "fill", and "fill-opacity". The fill and stroke color values should be specified as six-digit hex codes, and the opacity and width values should be supplied as floating-point numbers.

overlay_markers The prepared overlay markers (optional). See the function prep_overlay_markers for more information on how to specify a marker overlay.

longitude A vector of longitudes; inferred from the input dataset if data is provided.
latitude A vector of latitudes; inferred from the input dataset if data is provided.
zoom The map zoom. The map will infer this from the overlay unless longitude, latitude, and zoom are all specified.
width The map width; defaults to NULL
height The map height; defaults to NULL
bearing The map bearing; defaults to NULL
pitch The map pitch; defaults to NULL
scale ratio to scale the output image; ‘scale = 1’ will return the largest possible image. defaults to 0.5
scaling_factor The scaling factor of the tiles; either "1x" (the default) or "2x"
attribution Controls whether there is attribution on the image. Defaults to ‘TRUE’. Note: If attribution=false, the watermarked attribution is removed from the image. You still have a legal responsibility to attribute maps that use OpenStreetMap data, which includes most maps from Mapbox. If you specify attribution=false, you are legally required to include proper attribution elsewhere on the webpage or document.
logo Controls whether there is a Mapbox logo on the image. Defaults to ‘TRUE’.
before_layer A character string that specifies where in the hierarchy of layer elements the overlay should be inserted. The overlay will be placed just above the specified layer in the given Mapbox styles.
access_token A Mapbox access token; can be set with mb_access_token().
image If FALSE, return the a response() object from httr::GET() using the static image URL; defaults to TRUE
data An input data frame with longitude and latitude columns (X and Y or lon and lat as names are also acceptable) or an sf object with geometry type POINT.
marker_type The marker type; one of "pin-s", for a small pin; "pin-l", for a large pin; and "url", for an image path.
label The marker label (optional). Can be a letter, number (0 through 99), or a valid Maki icon (see https://labs.mapbox.com/maki-icons/) for options).
color The marker color (optional). Color should be specified as a three or six-digit hexadecimal code without the number sign.
url The URL of the image to be used for the icon if marker_type = "url".
Value

A pointer to an image of class "magick-image" if `image = TRUE`. The resulting image can be manipulated further with functions from the magick package.

A formatted list of marker specifications that can be passed to the `static_mapbox` function.

Examples

```r
## Not run:
library(mapboxapi)

points_of_interest <- tibble::tibble(
  longitude = c(-73.99405, -74.00616, -73.99577, -74.00761),
  latitude = c(40.72033, 40.72182, 40.71590, 40.71428)
)

prepped_pois <- prep_overlay_markers(
  data = points_of_interest,
  marker_type = "pin-l",
  label = 1:4,
  color = "fff"
)

map <- static_mapbox(
  style_id = "streets-v11",
  username = "mapbox",
  overlay_markers = prepped_pois,
  width = 1200,
  height = 800
)

map

## End(Not run)
```

---

tippecanoe  Generate an .mbtiles file with tippecanoe

Description

Generate an .mbtiles file with tippecanoe

Usage

tippecanoe(
  input,
  output,
  layer_name = NULL,
  ...)
Arguments

input  The dataset from which to generate vector tiles. Can be an sf object or GeoJSON file on disk.

output The name of the output .mbtiles file (with .mbtiles extension). Will be saved in the current working directory.

layer_name The name of the layer in the output .mbtiles file. If NULL, will either be a random string (if input is an sf object) or the name of the input GeoJSON file (if input is a file path).

min_zoom The minimum zoom level for which to compute tiles.

max_zoom The maximum zoom level for which to compute tiles. If both min_zoom and max_zoom are blank, tippecanoe will guess the best zoom levels for your data.

drop_rate The rate at which tippecanoe will drop features as you zoom out. If NULL, tippecanoe will drop features as needed in the densest tiles to stay within Mapbox’s limits.

overwrite If TRUE, an existing .mbtiles file with the same name will be overwritten.

other_options A character string of other options to be passed to the tippecanoe program.

keep_geojson Whether nor not to keep the temporary CSV or GeoJSON file used to generate the tiles. Defaults to FALSE.

Examples

## Not run:

# Workflow: create a dynamic tileset for dot-density mapping
library(tidycensus)
library(sf)
library(mapboxapi)

# Get population data for Census tracts in Vermont
vt_population <- get_decennial(
  geography = "tract",
  variables = "P001001",
  state = "Vermont",
  year = 2010,
  geometry = TRUE
)

# Convert to representative dots - 1 per person
vt_dots <- st_sample(
  min_zoom = NULL,
  max_zoom = NULL,
  drop_rate = NULL,
  overwrite = TRUE,
  other_options = NULL,
  keep_geojson = FALSE
)
upload_tiles

vt_population,
  size = vt_population$value
)

# Use tippecanoe to create dynamic tiles
tippecanoe(
  input = vt_dots,
  output = "vt_population.mbtiles",
  layer_name = "vermont_population",
  max_zoom = 18,
  drop_rate = 1.5
)

# Upload to your Mapbox account for visualization
# A Mapbox secret access token must be set with mb_access_token()
# to upload data to your account
upload_tiles(
  input = "vt_population.mbtiles",
  username = "kwalkertcu",
  tileset_id = "vt_population_dots",
  multipart = TRUE
)

## End(Not run)

---

upload_tiles

**Upload dataset to your Mapbox account**

### Description

Upload dataset to your Mapbox account

### Usage

```r
upload_tiles(
  input, username,
  access_token = NULL,
  tileset_id = NULL,
  tileset_name = NULL,
  keep_geojson = FALSE,
  multipart = FALSE
)
```

### Arguments

- **input**: An sf object, or the path to the dataset to upload as a character string.
- **username**: Your Mapbox username
access_token    Your Mapbox access token; must have secret scope
tileset_id      The ID of the tileset in your Mapbox account
tileset_name    The name of the tileset in your Mapbox account
keep_geojson    Whether or not to keep the temporary GeoJSON used to generate the tiles (if the
                input is an sf object)
multipart       Whether or not to upload to the temporary AWS staging bucket as a multipart
                object; defaults to FALSE.

Examples

## Not run:

# Example: create a tileset of median age for all United States Census tracts
# Requires setting a Mapbox secret access token as an environment variable

library(mapboxapi)
library(tidycensus)
options(tigris_use_cache = TRUE)

median_age <- get_acs(
  geography = "tract",
  variables = "B01002_001",
  state = c(state.abb, "DC"),
  geometry = TRUE
)

upload_tiles(
  input = median_age,
  username = "kwalkertcu", # Your username goes here
  tileset_id = "median_age",
  tileset_name = "us_median_age_2014_to_2018"
)

## End(Not run)
Index

addMapboxTiles, 2

check_upload_status, 3

gGet.mb_access_token (mb_access_token), 10
gGet.static_tiles, 4
gGet.style, 6
gGet.vector_tiles, 6

layer_spatial, 8
layer_static_mapbox, 7
list_styles, 9
list_tokens (mb_access_token), 10

mapboxapi, 9
mb_access_token, 10
mb_directions, 11
mb_geocode, 14
mb_isochrone, 16
mb_matrix, 18
mb_optimized_route, 19
mb_reverse_geocode (mb_geocode), 14

prep_overlay_markers, 8, 24
prep_overlay_markers (static_mapbox), 22

query_tiles, 21

static_mapbox, 22, 25

tippecanoe, 25

upload_tiles, 27