Package ‘mapboxapi’

October 5, 2020

Type Package

Title R Interface to ‘Mapbox’ Web Services

Date 2020-09-29

Version 0.2

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

LazyData true

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

Suggests mapdeck, tigris, tidycensus

RoxygenNote 7.1.1

NeedsCompilation no

Author Kyle Walker [aut, cre]

Repository CRAN

Date/Publication 2020-10-05 11:20:02 UTC

R topics documented:

  addMapboxTiles .................................................. 2
  check_upload_status ........................................... 3
  get_style ....................................................... 4
  get_vector_tiles ............................................... 4
Use a Mapbox style in a Leaflet map

Usage

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

Arguments

- **map**: A map widget object created by `leaflet::leaflet()`.
- **style_id**: The style ID of your Mapbox style.
- **username**: Your Mapbox username.
- **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.
check_upload_status

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()

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

Examples

## Not run:

```r
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

```r
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_style

Get information about a style from your Mapbox account

Description

Get information about a style from your Mapbox account

Usage

get_style(style_id, username, access_token = NULL)

Arguments

style_id The style ID
username Your Mapbox username
access_token Your 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

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

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)
```

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

---

**mb_access_token**

*Install a Mapbox access token in your .Renviron for repeated use*

**Description**

Install a Mapbox access token in your .Renviron for repeated use

List tokens from a Mapbox account

**Usage**

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

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

**Arguments**

- **token**
  The 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.

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

- **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.
The maximum number of tokens to return. Defaults to NULL.

How to sort the returned tokens; one of "created" or "modified".

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.

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

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")
## 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)
```

Make a request to the Mapbox Directions API

Make a request to the Mapbox Directions API

Usage

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

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.

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. 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

## 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
)

mb_reverse_geocode(
  coordinates,
  endpoint = "mapbox.places",
  limit = 1,
  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](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

```r
## 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)
```
mb_isochrone  
*Generate an isochrone using the Mapbox API*

**Description**

Generate an isochrone using the Mapbox API

**Usage**

```
mb_isochrone(
  location,  
  profile = "driving",  
  time = c(5, 10, 15),  
  access_token = NULL,  
  denoise = 1,  
  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", or "cycling". "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.
- **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.
- **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.
**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,  # 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.
  destinations = NULL,  # The destination coordinates of your request. If NULL (the default), a many-to-many matrix using origins will be returned.
  profile = "driving",  # The traffic profile for the request. Options include "driving", "walking", and "cycling".
  fallback_speed = NULL,  # The fallback speed to use if the requested profile is not available.
  access_token = NULL,  # Your Mapbox access token.
  duration_output = "minutes"  # The output duration units. Options include "minutes" and "hours".
)
```

### Arguments

- **origins**: 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.

- **destinations**: The destination coordinates of your request. If NULL (the default), a many-to-many matrix using origins will be returned.
mb_optimized_route

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

fallback_speed
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.

access_token
A Mapbox access token (required)

duration_output
one of "minutes" (the default) or "seconds"

Value
An R matrix of source-destination travel times.

Examples

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

philly_tracts <- tracts("PA", "Philadelphia", cb = TRUE, class = "sf")
downtown_philly <- mb_geocode("Philadelphia City Hall, Philadelphia PA")

time_to_downtown <- mb_matrix(philly_tracts, downtown_philly)

philly_tracts$time <- time_to_downtown

mapdeck(style = mapdeck_style("light")) %>%
  add_polygon(data = philly_tracts,
              fill_colour = "time",
              fill_opacity = 0.6,
              legend = TRUE)

## End(Not run)
```
Usage

mb_optimized_route(
  input_data,
  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.

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.
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

query_tiles(
  location,  # required
  tileset_id,  # required
  radius = 0,  # optional
  limit = 5,  # optional
)

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()

Value

Either a list of two sf objects - one representing the waypoints, and one representing the route - or an R list representing the full optimization API response.

Examples

## Not run:

library(mapboxapi)
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)
```r
query_tiles(
  location = NULL,
  tileset_id = NULL,
  radius = NULL,
  limit = NULL,
  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**
  Your 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**


**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

```r
static_mapbox(
  style_id, username, overlay_sf = NULL, overlay_style = NULL, overlay_markers = NULL, longitude = NULL, latitude = NULL, zoom = NULL, width = 600, height = 400, bearing = NULL, pitch = NULL, scaling_factor = c("1x", "2x"), before_layer = NULL, access_token = NULL
)

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

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>style_id</code></td>
<td>The style ID</td>
</tr>
<tr>
<td><code>username</code></td>
<td>Your Mapbox username</td>
</tr>
</tbody>
</table>
**static_mapbox**

- **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 600px

- **height**: The map height; defaults to 600px

- **bearing**: The map bearing; defaults to 0

- **pitch**: The map pitch; defaults to 0

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

- **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**: Your Mapbox access token; can be set with `mb_access_token()`.

- **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/](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". 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

```r
tippecanoe(
  input,
  output,
  layer_name = NULL,
  min_zoom = NULL,
  max_zoom = NULL,
  drop_rate = NULL,
  overwrite = TRUE,
  other_options = 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(
  vt_population,
  size = vt_population$value
)

# Use tippecanoe to create dynamic tiles
upload_tiles

Description

Upload dataset to your Mapbox account

Usage

```r
upload_tiles(
  input,  # An sf object, or the path to the dataset to upload as a character string.
  username,  # Your Mapbox username
  access_token = NULL,  # Your Mapbox access token; must have secret scope
  tileset_id = NULL,  # The ID of the tileset in your Mapbox account
  tileset_name = NULL,  # The name of the tileset in your Mapbox account
  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
get_style, 4
get_vector_tiles, 4
list_styles, 5
list_tokens (mb_access_token), 6
mapboxapi, 6
mb_access_token, 6
mb_directions, 7
mb_geocode, 10
mb_isochrone, 12
mb_matrix, 13
mb_optimized_route, 14
mb_reverse_geocode (mb_geocode), 10
prep_overlay_markers, 19
prep_overlay_markers (static_mapbox), 18
query_tiles, 16
static_mapbox, 18, 19
tippecanoe, 20
upload_tiles, 22