

Package ‘googletraffic’

July 5, 2023

Title Google Traffic

Version 0.1.4

Description

Create geographically referenced traffic data from the Google Maps JavaScript API <<https://developers.google.com/maps/documentation/javascript/examples/layer-traffic>>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.1

URL <https://dime-worldbank.github.io/googletraffic/>

BugReports <https://github.com/dime-worldbank/googletraffic/issues>

Imports dplyr, googleway, htmlwidgets, plotwidgets, png, sf, sp, stringr, webshot2, raster, ColorNameR, schemr

NeedsCompilation no

Author Robert Marty [aut, cre] (<<https://orcid.org/0000-0002-3164-3813>>)

Maintainer Robert Marty <rmarty@worldbank.org>

Repository CRAN

Date/Publication 2023-07-05 05:00:02 UTC

R topics documented:

gt_load_png_as_traffic_raster	2
gt_make_grid	3
gt_make_png	5
gt_make_raster	6
gt_make_raster_from_grid	8
gt_make_raster_from_polygon	10
gt_mosaic	12

Index	13
--------------	-----------

```
gt_load_png_as_traffic_raster
    Converts PNG to raster
```

Description

Converts PNG of **Google traffic data** to raster and translates color values to traffic values

Usage

```
gt_load_png_as_traffic_raster(
    filename,
    location,
    height,
    width,
    zoom,
    traffic_color_dist_thresh = 4.6,
    traffic_color_dist_metric = "CIEDE2000"
)
```

Arguments

filename	Filename of PNG file
location	Vector of latitude and longitude used to create PNG file using gt_make_png()
height	Height (in pixels; pixel length depends on zoom) used to create PNG file using gt_make_png()
width	Width (in pixels; pixel length depends on zoom) used to create PNG file using gt_make_png()
zoom	Zoom level used to create PNG file using gt_make_png()
traffic_color_dist_thresh	Google traffic relies on four main base colors: #63D668 for no traffic, #FF974D for medium traffic, #F23C32 for high traffic, and #811F1F for heavy traffic. Slight variations of these colors can also represent traffic. By default, the base colors and all colors within a 4.6 color distance of each base color are used to define traffic; by default, the CIEDE2000 metric is used to determine color distance. A value of 2.3 is one threshold used to define a "just noticeable distance" (JND) between colors (by default, 2 X JND is used). This parameter changes the color distance from the base colors used to define colors as traffic. For more information, see here .
traffic_color_dist_metric	See above; this parameter changes the metric used to calculate distances between colors. By default, CIEDE2000 is used; CIE76 and CIE94 can also be used. For more information, see here .

Value

Returns a raster where each pixel represents traffic level (1 = no traffic, 2 = medium traffic, 3 = traffic delays, 4 = heavy traffic)

References

Markus Hilpert, Jenni A. Shearston, Jemaleddin Cole, Steven N. Chillrud, and Micaela E. Martinez. [Acquisition and analysis of crowd-sourced traffic data](#). CoRR, abs/2105.12235, 2021.

Pavel Pokorny. [Determining traffic levels in cities using google maps](#). In 2017 Fourth International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), pages 144–147, 2017.

Examples

```
## Not run:
## Make png
gt_make_png(location      = c(40.712778, -74.006111),
            height       = 1000,
            width        = 1000,
            zoom         = 16,
            out_filename = "google_traffic.png",
            google_key   = "GOOGLE-KEY-HERE")

## Load png as traffic raster
r <- gt_load_png_as_traffic_raster(filename = "google_traffic.png",
                                  location = c(40.712778, -74.006111),
                                  height   = 1000,
                                  width    = 1000,
                                  zoom     = 16)

## End(Not run)
```

gt_make_grid

Creates Grid to Query Google Traffic

Description

Creates a grid of sf polygons, where traffic data for each polygon can then be queried using [gt_make_raster_from_grid\(\)](#).

Usage

```
gt_make_grid(
  polygon,
  zoom,
  height_width_max = 2000,
  height = NULL,
```

```

width = NULL,
reduce_hw = 10
)

```

Arguments

polygon	Polygon (sf object or SpatialPolygonsDataframe) in WGS84 CRS the defines region to be queried.
zoom	Zoom level; integer from 5 to 20. For more information about how zoom levels correspond to pixel size, see here and here .
height_width_max	Maximum pixel height and width to check using for each grid (pixel length depends on zoom). If the same number of grids can be made with a smaller height/width, the function will use a smaller height/width. If height and width are specified, that height and width will be used and height_width_max will be ignored. (Default: 2000)
height	Height, in pixels, for each grid (pixel length depends on zoom). Enter a height to manually specify the height; otherwise, a height of height_width_max or smaller will be used.
width	Pixel, in pixels, for each grid (pixel length depends on zoom). Enter a width to manually specify the width; otherwise, a width of height_width_max or smaller will be used.
reduce_hw	Number of pixels to reduce height/width by. Doing so creates some overlap between grids to ensure there is not blank space between grids. (Default: 10).

Value

Returns an sf dataframe with the locations to query, including parameters needed for [gt_make_raster_from_grid\(\)](#)

Examples

```

## Make polygon
poly_sf <- c(xmin = -74.02426,
             xmax = -73.91048,
             ymin = 40.70042,
             ymax = 40.87858) |>
  sf::st_bbox() |>
  sf::st_as_sf() |>
  sf::st_as_sf()

sf::st_crs(poly_sf) <- 4326

## Make grid using polygon
grid_sf <- gt_make_grid(polygon = poly_sf,
                       height = 2000,
                       width = 2000,
                       zoom = 16)

```

`gt_make_png`*Make Google Traffic PNG*

Description

Make a png file of **Google traffic data**. The `gt_load_png_as_traffic_raster()` function can then be used to convert the png into a traffic raster

Usage

```
gt_make_png(  
    location,  
    height,  
    width,  
    zoom,  
    out_filename,  
    google_key,  
    webshot_zoom = 1,  
    webshot_delay = NULL,  
    print_progress = TRUE  
)
```

Arguments

<code>location</code>	Vector of latitude and longitude
<code>height</code>	Height (in pixels; pixel length depends on zoom)
<code>width</code>	Width (in pixels; pixel length depends on zoom)
<code>zoom</code>	Zoom level; integer from 5 to 20. For more information about how zoom levels correspond to pixel size, see here and here .
<code>out_filename</code>	Filename of PNG file to make
<code>google_key</code>	Google API key, where the Maps JavaScript API is enabled. To create a Google API key, follow these instructions .
<code>webshot_zoom</code>	How many pixels should be created relative to height and width values. If height and width are set to 100 and <code>webshot_zoom</code> is set to 2, the resulting raster will have dimensions of about 200x200 (default: 1).
<code>webshot_delay</code>	How long to wait for Google traffic layer to render. Larger height/widths require longer delay times. If NULL, the following delay time (in seconds) is used: $\text{delay} = \max(\text{height}, \text{width})/200$.
<code>print_progress</code>	Whether to print function progress (default: TRUE)

Value

Returns a PNG file showing traffic levels.

References

Markus Hilpert, Jenni A. Shearston, Jemaleddin Cole, Steven N. Chillrud, and Micaela E. Martinez. [Acquisition and analysis of crowd-sourced traffic data](#). CoRR, abs/2105.12235, 2021.

Pavel Pokorny. [Determining traffic levels in cities using google maps](#). In 2017 Fourth International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), pages 144–147, 2017.

Examples

```
## Not run:
gt_make_png(location    = c(40.712778, -74.006111),
            height     = 1000,
            width      = 1000,
            zoom       = 16,
            out_filename = "google_traffic.png",
            google_key  = "GOOGLE-KEY-HERE")

## End(Not run)
```

gt_make_raster	<i>Make Google Traffic Raster</i>
----------------	-----------------------------------

Description

Make a raster of [Google traffic data](#), where each pixel has one of four values indicating traffic volume (no traffic, light, moderate, and heavy).

Usage

```
gt_make_raster(
  location,
  height,
  width,
  zoom,
  google_key,
  traffic_color_dist_thresh = 4.6,
  traffic_color_dist_metric = "CIEDE2000",
  webshot_zoom = 1,
  webshot_delay = NULL,
  print_progress = TRUE
)
```

Arguments

location	Vector of latitude and longitude
height	Height (in pixels; pixel length depends on zoom)

width	Width (in pixels; pixel length depends on zoom)
zoom	Zoom level; integer from 5 to 20. For more information about how zoom levels correspond to pixel size, see here and here .
google_key	Google API key, where the Maps JavaScript API is enabled. To create a Google API key, follow these instructions .
traffic_color_dist_thresh	Google traffic relies on four main base colors: #63D668 for no traffic, #FF974D for medium traffic, #F23C32 for high traffic, and #811F1F for heavy traffic. Slight variations of these colors can also represent traffic. By default, the base colors and all colors within a 4.6 color distance of each base color are used to define traffic; by default, the CIEDE2000 metric is used to determine color distance. A value of 2.3 is one threshold used to define a "just noticeable distance" (JND) between colors (by default, 2 X JND is used). This parameter changes the color distance from the base colors used to define colors as traffic. For more information, see here .
traffic_color_dist_metric	See above; this parameter changes the metric used to calculate distances between colors. By default, CIEDE2000 is used; CIE76 and CIE94 can also be used. For more information, see here .
webshot_zoom	How many pixels should be created relative to height and width values. If height and width are set to 100 and webshot_zoom is set to 2, the resulting raster will have dimensions of about 200x200 (default: 1).
webshot_delay	How long to wait for Google traffic layer to render. Larger height/widths require longer delay times. If NULL, the following delay time (in seconds) is used: $\text{delay} = \max(\text{height}, \text{width}) / 200$.
print_progress	Whether to print function progress (default: TRUE)

Value

Returns a georeferenced raster. Raster pixels can contain the following values: 1 = no traffic; 2 = medium traffic; 3 = high traffic; 4 = heavy traffic.

References

Markus Hilpert, Jenni A. Shearston, Jemaleddin Cole, Steven N. Chillrud, and Micaela E. Martinez. [Acquisition and analysis of crowd-sourced traffic data](#). CoRR, abs/2105.12235, 2021.

Pavel Pokorny. [Determining traffic levels in cities using google maps](#). In 2017 Fourth International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), pages 144–147, 2017.

Examples

```
## Not run:
r <- gt_make_raster(location = c(40.712778, -74.006111),
                    height   = 1000,
                    width    = 1000,
                    zoom      = 16,
```

```

google_key = "GOOGLE-KEY-HERE")

## End(Not run)

```

```
gt_make_raster_from_grid
```

Make Google Traffic Raster Based on Grid of Coordinates

Description

Make a raster of **Google traffic data**, where each pixel has one of four values indicating traffic volume (no traffic, light, moderate, and heavy).

Usage

```

gt_make_raster_from_grid(
  grid_param_df,
  google_key,
  traffic_color_dist_thresh = 4.6,
  traffic_color_dist_metric = "CIEDE2000",
  webshot_zoom = 1,
  webshot_delay = NULL,
  return_list_of_rasters = FALSE,
  print_progress = TRUE
)

```

Arguments

`grid_param_df` Grid parameter dataframe produced from `gt_make_grid()`

`google_key` Google API key, where the **Maps JavaScript API** is enabled. To create a Google API key, follow [these instructions](#).

`traffic_color_dist_thresh`
 Google traffic relies on four main base colors: #63D668 for no traffic, #FF974D for medium traffic, #F23C32 for high traffic, and #811F1F for heavy traffic. Slight variations of these colors can also represent traffic. By default, the base colors and all colors within a 4.6 color distance of each base color are used to define traffic; by default, the CIEDE2000 metric is used to determine color distance. A value of 2.3 is one threshold used to define a "just noticeable distance" (JND) between colors (by default, 2 X JND is used). This parameter changes the color distance from the base colors used to define colors as traffic. For more information, see [here](#).

`traffic_color_dist_metric`
 See above; this parameter changes the metric used to calculate distances between colors. By default, CIEDE2000 is used; CIE76 and CIE94 can also be used. For more information, see [here](#).

`webshot_zoom` How many pixels should be created relative to height and width values. If height and width are set to 100 and `webshot_zoom` is set to 2, the resulting raster will have dimensions of about 200x200 (default: 1).

`webshot_delay` How long to wait for Google traffic layer to render. Larger height/widths require longer delay times. If NULL, the following delay time (in seconds) is used: $\text{delay} = \max(\text{height}, \text{width}) / 200$.

`return_list_of_rasters` Instead of merging traffic rasters produced for each grid together into one large raster, return a list of rasters (default: FALSE)

`print_progress` Whether to print function progress (default: TRUE)

Value

Returns a georeferenced raster. Raster pixels can contain the following values: 1 = no traffic; 2 = medium traffic; 3 = high traffic; 4 = heavy traffic.

References

Markus Hilpert, Jenni A. Shearston, Jemaleddin Cole, Steven N. Chillrud, and Micaela E. Martinez. [Acquisition and analysis of crowd-sourced traffic data](#). CoRR, abs/2105.12235, 2021.

Pavel Pokorny. [Determining traffic levels in cities using google maps](#). In 2017 Fourth International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), pages 144–147, 2017.

Examples

```
## Not run:
## Grab polygon of Manhattan
us_sp <- raster::getData('GADM', country='USA', level=2)
ny_sp <- us_sp[us_sp$NAME_2 %in% "New York",]

## Make Grid
grid_df <- gt_make_grid(polygon = ny_sp,
                       height   = 2000,
                       width    = 2000,
                       zoom     = 16)

## Make raster from grid
r <- gt_make_raster_from_grid(grid_param_df = grid_clean_df,
                             google_key   = "GOOGLE-KEY-HERE")

## End(Not run)
```

`gt_make_raster_from_polygon`*Make Google Traffic Raster Based on Polygon*

Description

Make a raster of **Google traffic data**, where each pixel has one of four values indicating traffic volume (no traffic, light, moderate, and heavy).

Usage

```
gt_make_raster_from_polygon(  
  polygon,  
  zoom,  
  google_key,  
  height_width_max = 2000,  
  height = NULL,  
  width = NULL,  
  traffic_color_dist_thresh = 4.6,  
  traffic_color_dist_metric = "CIEDE2000",  
  webshot_zoom = 1,  
  webshot_delay = NULL,  
  reduce_hw = 10,  
  return_list_of_rasters = FALSE,  
  mask_to_polygon = TRUE,  
  print_progress = TRUE  
)
```

Arguments

<code>polygon</code>	Polygon (sf object or SpatialPolygonsDataframe) in WGS84 CRS
<code>zoom</code>	Zoom level; integer from 5 to 20. For more information about how zoom levels correspond to pixel size, see here and here .
<code>google_key</code>	Google API key, where the Maps JavaScript API is enabled. To create a Google API key, follow these instructions .
<code>height_width_max</code>	Maximum pixel height and width to check using for each API query (pixel length depends on zoom). If the same number of API queries can be made with a smaller height/width, the function will use a smaller height/width. If height and width are specified, that height and width will be used and <code>height_width_max</code> will be ignored. (Default: 2000)
<code>height</code>	Height, in pixels, for each API query (pixel length depends on zoom). Enter a height to manually specify the height; otherwise, a height of <code>height_width_max</code> or smaller will be used.

width	Pixel, in pixels, for each API query (pixel length depends on zoom). Enter a width to manually specify the width; otherwise, a width of height_width_max or smaller will be used.
traffic_color_dist_thresh	Google traffic relies on four main base colors: #63D668 for no traffic, #FF974D for medium traffic, #F23C32 for high traffic, and #811F1F for heavy traffic. Slight variations of these colors can also represent traffic. By default, the base colors and all colors within a 4.6 color distance of each base color are used to define traffic; by default, the CIEDE2000 metric is used to determine color distance. A value of 2.3 is one threshold used to define a "just noticeable distance" (JND) between colors (by default, 2 X JND is used). This parameter changes the color distance from the base colors used to define colors as traffic. For more information, see here .
traffic_color_dist_metric	See above; this parameter changes the metric used to calculate distances between colors. By default, CIEDE2000 is used; CIE76 and CIE94 can also be used. For more information, see here .
webshot_zoom	How many pixels should be created relative to height and width values. If height and width are set to 100 and webshot_zoom is set to 2, the resulting raster will have dimensions of about 200x200 (default: 1).
webshot_delay	How long to wait for Google traffic layer to render (in seconds). Larger height/widths require longer delay times. If NULL, the following delay time (in seconds) is used: $\text{delay} = \max(\text{height}, \text{width}) / 200$.
reduce_hw	Number of pixels to reduce height/width by. Doing so creates some overlap between grids to ensure there is not blank space between tiles. (Default: 10).
return_list_of_rasters	Whether to return a list of raster tiles instead of mosaicing together. (Default: FALSE).
mask_to_polygon	Whether to mask raster to polygon. (Default: TRUE).
print_progress	Show progress for which grid / API query has been processed. (Default: TRUE).

Value

Returns a georeferenced raster. Raster pixels can contain the following values: 1 = no traffic; 2 = medium traffic; 3 = high traffic; 4 = heavy traffic.

References

Markus Hilpert, Jenni A. Shearston, Jemaleddin Cole, Steven N. Chillrud, and Micaela E. Martinez. [Acquisition and analysis of crowd-sourced traffic data](#). CoRR, abs/2105.12235, 2021.

Pavel Pokorny. [Determining traffic levels in cities using google maps](#). In 2017 Fourth International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), pages 144–147, 2017.

Examples

```
## Not run:
## Grab polygon of Manhattan
us_sp <- raster::getData('GADM', country='USA', level=2)
ny_sp <- us_sp[us_sp$NAME_2 %in% "New York",]

## Make raster
r <- gt_make_raster_from_polygon(polygon = ny_sp,
                                height  = 2000,
                                width   = 2000,
                                zoom     = 16,
                                google_key = "GOOGLE-KEY-HERE")

## End(Not run)
```

gt_mosaic

Mosaic rasters with different origins and resolutions

Description

The `raster::mosaic()` function requires rasters to have the same origin and resolution. However, when producing multiple rasters to query traffic data across a large study area, the rasters will not have the same origins and may not have the same resolutions (in cases where rasters at different latitudes are queried). `gt_mosaic()` allows for mosaicing rasters with different origins and resolutions.

Usage

```
gt_mosaic(r_list)
```

Arguments

`r_list` List of rasters

Value

Returns a raster.

Examples

```
r1 <- raster::raster(ncol=10, nrow=10, xmn = -10, xmx = 1, ymn = -10, ymx = 1)
r2 <- raster::raster(ncol=10, nrow=10, xmn = 0, xmx = 10, ymn = 0, ymx = 10)
r3 <- raster::raster(ncol=10, nrow=10, xmn = 9, xmx = 20, ymn = 9, ymx = 20)

r123 <- list(r1, r2, r3)

r <- gt_mosaic(r123)
```

Index

`gt_load_png_as_traffic_raster`, [2](#)
`gt_load_png_as_traffic_raster()`, [5](#)
`gt_make_grid`, [3](#)
`gt_make_grid()`, [8](#)
`gt_make_png`, [5](#)
`gt_make_png()`, [2](#)
`gt_make_raster`, [6](#)
`gt_make_raster_from_grid`, [8](#)
`gt_make_raster_from_grid()`, [3](#), [4](#)
`gt_make_raster_from_polygon`, [10](#)
`gt_mosaic`, [12](#)

`raster::mosaic()`, [12](#)