Package 'mapdeck'

August 16, 2018

Type Package

Title Interactive Maps Using 'Mapbox GL JS' and 'Deck.gl'

Version 0.1.0

Date 2018-08-11

Description Provides a mechanism to plot an interactive map using 'Mapbox GL' (<https://www.mapbox.com/mapbox-gl-js/api/>), a javascript library for interactive maps, and 'Deck.gl' (<http://deck.gl/#/>), a javascript library which uses 'WebGL' for visualising large data sets.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.3.0)

Imports googlePolylines (>= 0.6.2), htmlwidgets, htmltools, scales, shiny, jsonlite, magrittr, viridisLite

RoxygenNote 6.0.1

Suggests covr, geojsonsf, knitr, rmarkdown, sf, testthat

VignetteBuilder knitr

NeedsCompilation no

Author David Cooley [aut, cre]

Maintainer David Cooley <dcooley@symbolix.com.au>

Repository CRAN

Date/Publication 2018-08-16 10:30:05 UTC

R topics documented:

`add_arc` *Add arc*

Description

The Arc Layer renders raised arcs joining pairs of source and target coordinates

Usage

```
add_arc(map, data = get_map_data(map), layer_id, origin, destination,
  id = NULL, stroke_from = NULL, stroke_from_opacity = NULL,
  stroke_to = NULL, stroke_to_opacity = NULL, stroke_width = NULL,
  tooltip = NULL, digits = 6, palette = viridisLite::viridis)
```

Arguments

`map`	a mapdeck map object
`data`	data to be used in the layer
`layer_id`	single value specifying an id for the layer. Use this value to distinguish between shape layers of the same type
`origin`	vector of longitude and latitude columns, or an `sfc` column
`destination`	vector of longitude and latitude columns, or an `sfc` column
`id`	an id value in `data` to identify layers when interacting in Shiny apps

stroke_from	variable or hex colour to use as the staring stroke colour
stroke_from_opacity	value between 1 and 255. Either a string specifying the column of data containing the stroke opacity of each shape, or a value between 1 and 255 to be applied to all the shapes
stroke_to	variable or hex colour to use as the ending stroke colour
stroke_to_opacity	value between 1 and 255. Either a string specifying the column of data containing the stroke opacity of each shape, or a value between 1 and 255 to be applied to all the shapes
stroke_width	width of the stroke
tooltip	variable of data containing text or HTML to render as a tooltip
digits	integer. Use this parameter to specify how many digits (decimal places) should be used for the latitude / longitude coordinates.
palette	a function which generates hex colours

Details

MULTIPOINT objects will be treated as single points. That is, if an sf objet has one row with a MULTIPOINT object consisting of two points, this will be expanded to two rows of single POINTs. Therefore, if the origin is a MULTIPOINT of two points, and the destination is a single POINT, the code will error as there will be an uneven number of rows

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

url <- 'https://raw.githubusercontent.com/plotly/datasets/master/2011_february_aa_flight_paths.csv'
flights <- read.csv(url)
flights$id <- seq_len(nrow(flights))
flights$stroke <- sample(1:3, size = nrow(flights), replace = T)
flights$info <- paste0("<b>",flights$airport1, " - ", flights$airport2, "</b>")

mapdeck( token = key, style = 'mapbox://styles/mapbox/dark-v9', pitch = 45 ) %>%
  add_arc(
  data = flights
  , layer_id = "arc_layer"
  , origin = c("start_lon", "start_lat")
  , destination = c("end_lon", "end_lat")
  , stroke_from = "airport1"
  , stroke_to = "airport2"
  , stroke_width = "stroke"
  , tooltip = "info"
 )

## Using a 2-sfc-column sf object
```

```
library(sf)

sf_flights <- cbind(
  sf::st_as_sf(flights, coords = c("start_lon", "start_lat"))
  , sf::st_as_sf(flights[, c("end_lon","end_lat")], coords = c("end_lon", "end_lat"))
)

mapdeck(
  token = key
) %>%
 add_arc(
   data = sf_flights
   , origin = 'geometry'
   , destination = 'geometry.1'
   , layer_id = 'arcs'
)
```

---

add_geojson *Add Geojson*

Description

The GeoJson Layer takes in GeoJson formatted data and renders it as interactive polygons, lines and points

Usage

```
add_geojson(map, data = get_map_data(map), layer_id, lineColor = "#440154",
  fillColor = "#440154", radius = 1, lineWidth = 1,
  light_settings = list(), elevation = 0)
```

Arguments

map	a mapdeck map object
data	data to be used in the layer
layer_id	single value specifying an id for the layer. Use this value to distinguish between shape layers of the same type
lineColor	hex value for all line colours. See details
fillColor	hex value for all fill colours. See details
radius	radius of points in meters. See details
lineWidth	width of lines in meters. See details
light_settings	list of light setting parameters. See light_settings
elevation	elevation of polygons. See details

Details

The GeoJSON string needs to have a `class` attribute of 'json'

If the GeoJSON contains the following fields in the `properties` object, they will be used as the attribute properties for each feature. Otherwise the values supplied to the arguments will be applied to all the features.

- fillColor - fill colour of polygons and points
- lineColor - line colour of lines
- lineWidth - line width of lines
- elevation - elevation of polygons
- radius - radius of points

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

mapdeck(
 token = key
 , location = c(145, -37.9)
 , zoom = 8
 , style = "mapbox://styles/mapbox/dark-v9"
 , pitch = 35
) %>%
 add_geojson(
   data = geojson
   , layer_id = "geojson"
 )

## add colours, elevation and opacities
sf <- geojsonsf::geojson_sf(geojson)
sf$elevation <- sample(100:1000, size = nrow(sf), replace = T)
sf$fillOpacity <- sample(200:255, size = nrow(sf), replace = T)
sf$radius <- sample(1:100, size = nrow(sf), replace = T)

mapdeck(
  token = key
  , location = c(145, -37.9)
  , zoom = 8
  , style = "mapbox://styles/mapbox/dark-v9"
  , pitch = 35
) %>%
  add_geojson(
    data = sf
    , lineWidth = 250,
    , layer_id = "geojson"
 )
```

add_grid *Add Grid*

Description

The Grid Layer renders a grid heatmap based on an array of points. It takes the constant size all each cell, projects points into cells. The color and height of the cell is scaled by number of points it contains.

Usage

```
add_grid(map, data = get_map_data(map), lon = NULL, lat = NULL,
  polyline = NULL, colour_range = viridisLite::viridis(5),
  cell_size = 1000, extruded = TRUE, elevation_scale = 1, layer_id,
  digits = 6)
```

Arguments

map	a mapdeck map object
data	data to be used in the layer
lon	column containing longitude values
lat	column containing latitude values
polyline	column of data containing the polylines
colour_range	vector of hex colours
cell_size	size of each cell in meters
extruded	logical indicating if cells are elevated or not
elevation_scale	cell elevation multiplier
layer_id	single value specifying an id for the layer. Use this value to distinguish between shape layers of the same type
digits	integer. Use this parameter to specify how many digits (decimal places) should be used for the latitude / longitude coordinates.

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

df <- read.csv(paste0(
'https://raw.githubusercontent.com/uber-common/deck.gl-data/master/',
'examples/3d-heatmap/heatmap-data.csv'
```

```
))


mapdeck( token = key, style = 'mapbox://styles/mapbox/dark-v9', pitch = 45 ) %>%
add_grid(
  data = df
  , lat = "lat"
  , lon = "lng"
  , cell_size = 5000
  , elevation_scale = 50
  , layer_id = "grid_layer"
)
```

`add_line` *Add line*

Description

The Line Layer renders raised lines joining pairs of source and target coordinates

Usage

```
add_line(map, data = get_map_data(map), layer_id, origin, destination,
  id = NULL, stroke_colour = NULL, stroke_width = NULL,
  stroke_opacity = NULL, tooltip = NULL, digits = 6,
  palette = viridisLite::viridis)
```

Arguments

`map`	a mapdeck map object
`data`	data to be used in the layer
`layer_id`	single value specifying an id for the layer. Use this value to distinguish between shape layers of the same type
`origin`	vector of longitude and latitude columns, or an `sfc` column
`destination`	vector of longitude and latitude columns, or an `sfc` column
`id`	an id value in `data` to identify layers when interacting in Shiny apps
`stroke_colour`	variable or hex colour to use as the ending stroke colour
`stroke_width`	width of the stroke
`stroke_opacity`	value between 1 and 255. Either a string specifying the column of `data` containing the stroke opacity of each shape, or a value between 1 and 255 to be applied to all the shapes
`tooltip`	variable of `data` containing text or HTML to render as a tooltip
`digits`	integer. Use this parameter to specify how many digits (decimal places) should be used for the latitude / longitude coordinates.
`palette`	a function which generates hex colours

Details

MULTIPOINT objects will be treated as single points. That is, if an sf objet has one row with a MULTIPOINT object consisting of two points, this will be expanded to two rows of single POINTs. Therefore, if the origin is a MULTIPOINT of two points, and the destination is a single POINT, the code will error as there will be an uneven number of rows

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

url <- 'https://raw.githubusercontent.com/plotly/datasets/master/2011_february_aa_flight_paths.csv'
flights <- read.csv(url)
flights$id <- seq_len(nrow(flights))
flights$stroke <- sample(1:3, size = nrow(flights), replace = T)

mapdeck( token = key, style = 'mapbox://styles/mapbox/dark-v9', pitch = 45 ) %>%
  add_line(
    data = flights
    , layer_id = "line_layer"
    , origin = c("start_lon", "start_lat")
    , destination = c("end_lon", "end_lat")
    , stroke_colour = "airport1"
    , stroke_width = "stroke"
 )
```

---

add_path *Add Path*

Description

The Path Layer takes in lists of coordinate points and renders them as extruded lines with mitering.

Usage

```
add_path(map, data = get_map_data(map), polyline = NULL,
  stroke_colour = NULL, stroke_width = NULL, stroke_opacity = NULL,
  tooltip = NULL, layer_id, digits = 6, palette = viridisLite::viridis)
```

Arguments

map	a mapdeck map object
data	data to be used in the layer
polyline	column of data containing the polylines

stroke_colour	variable of data or hex colour for the stroke
stroke_width	width of the stroke
stroke_opacity	value between 1 and 255. Either a string specifying the column of data containing the stroke opacity of each shape, or a value between 1 and 255 to be applied to all the shapes
tooltip	variable of data containing text or HTML to render as a tooltip
layer_id	single value specifying an id for the layer. Use this value to distinguish between shape layers of the same type
digits	integer. Use this parameter to specify how many digits (decimal places) should be used for the latitude / longitude coordinates.
palette	a function which generates hex colours

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

mapdeck(
  token = key
  , style = 'mapbox://styles/mapbox/dark-v9'
  , location = c(145, -37.8)
  , zoom = 10) %>%
  add_path(
    data = roads
    , stroke_colour = "RIGHT_LOC"
    , layer_id = "path_layer"
    , tooltip = "ROAD_NAME"
  )
```

---

add_pointcloud *Add Pointcloud*

Description

The Pointcloud Layer takes in coordinate points and renders them as circles with a certain radius.

Usage

```
add_pointcloud(map, data = get_map_data(map), lon = NULL, lat = NULL,
  elevation, polyline = NULL, radius = NULL, fill_colour = NULL,
  fill_opacity = NULL, stroke_width = NULL, light_settings = list(),
  layer_id, digits = 6, palette = viridisLite::viridis)
```

Arguments

map	a mapdeck map object
data	data to be used in the layer
lon	column containing longitude values
lat	column containing latitude values
elevation	column containing the elevation values
polyline	column of data containing the polylines
radius	in metres
fill_colour	column of data or hex colour for the fill colour
fill_opacity	value between 1 and 255. Either a string specifying the column of data containing the fill opacity of each shape, or a value between 1 and 255 to be applied to all the shapes
stroke_width	width of the stroke
light_settings	list of light setting parameters. See light_settings
layer_id	single value specifying an id for the layer. Use this value to distinguish between shape layers of the same type
digits	integer. Use this parameter to specify how many digits (decimal places) should be used for the latitude / longitude coordinates.
palette	a function which generates hex colours

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

df <- capitals
df$z <- sample(10000:10000000, size = nrow(df))

mapdeck(token = key, style = 'mapbox://styles/mapbox/dark-v9') %>%
add_pointcloud(
  data = df
  , lon = 'lon'
  , lat = 'lat'
  , elevation = 'z'
  , layer_id = 'point'
  , fill_colour = "country"
)
```

add_polygon *Add Polygon*

Description

The Polygon Layer renders filled and/or stroked polygons.

Usage

```
add_polygon(map, data = get_map_data(map), polyline = NULL,
  stroke_colour = NULL, stroke_width = NULL, fill_colour = NULL,
  fill_opacity = NULL, elevation = NULL, tooltip = NULL,
  light_settings = list(), layer_id, digits = 6,
  palette = viridisLite::viridis)
```

Arguments

map	a mapdeck map object
data	data to be used in the layer
polyline	column of data containing the polylines
stroke_colour	variable of data or hex colour for the stroke
stroke_width	width of the stroke
fill_colour	column of data or hex colour for the fill colour
fill_opacity	value between 1 and 255. Either a string specifying the column of data containing the fill opacity of each shape, or a value between 1 and 255 to be applied to all the shapes
elevation	the height of the polygon
tooltip	variable of data containing text or HTML to render as a tooltip
light_settings	list of light setting parameters. See light_settings
layer_id	single value specifying an id for the layer. Use this value to distinguish between shape layers of the same type
digits	integer. Use this parameter to specify how many digits (decimal places) should be used for the latitude / longitude coordinates.
palette	a function which generates hex colours

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

df <- melbourne
df$elevation <- sample(100:5000, size = nrow(df))
```

```
df$info <- paste0("<b>SA2 - </b><br>",df$SA2_NAME)

mapdeck(
  token = key
  , style = mapdeck_style('dark')
  , location = c(145, -38)
  , zoom = 8
  ) %>%
  add_polygon(
    data = df
    , polyline = "geometry"
    , layer = "polygon_layer"
    , fill_colour = "fillColor",
    , stroke_colour = "fillColor",
    , elevation = "elevation"
    , stroke_width = 0
    , tooltip = 'info'
  )

library(sf)
library(geojsonsf)

sf <- geojson_sf("https://symbolixau.github.io/data/geojson/SA2_2016_VIC.json")

mapdeck(
  token = key
  , style = 'mapbox://styles/mapbox/dark-v9'
) %>%
  add_polygon(
    data = sf
    , layer = "polygon_layer"
    , fill_colour = "SA2_NAME16"
  )
```

`add_scatterplot` *Add Scatterplot*

Description

The Scatterplot Layer takes in coordinate points and renders them as circles with a certain radius.

Usage

```
add_scatterplot(map, data = get_map_data(map), lon = NULL, lat = NULL,
  polyline = NULL, radius = NULL, fill_colour = NULL,
  fill_opacity = NULL, layer_id, digits = 6,
  palette = viridisLite::viridis)
```

Arguments

map	a mapdeck map object
data	data to be used in the layer
lon	column containing longitude values
lat	column containing latitude values
polyline	column of data containing the polylines
radius	in metres
fill_colour	column of data or hex colour for the fill colour
fill_opacity	value between 1 and 255. Either a string specifying the column of data containing the fill opacity of each shape, or a value between 1 and 255 to be applied to all the shapes
layer_id	single value specifying an id for the layer. Use this value to distinguish between shape layers of the same type
digits	integer. Use this parameter to specify how many digits (decimal places) should be used for the latitude / longitude coordinates.
palette	a function which generates hex colours

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

mapdeck( token = key, style = 'mapbox://styles/mapbox/dark-v9', pitch = 45 ) %>%
add_scatterplot(
  data = capitals
  , lat = "lat"
  , lon = "lon"
  , radius = 100000
  , fill_colour = "country"
  , layer_id = "scatter_layer"
)

df <- read.csv(paste0(
'https://raw.githubusercontent.com/uber-common/deck.gl-data/master/',
'examples/3d-heatmap/heatmap-data.csv'
))

mapdeck( token = key, style = 'mapbox://styles/mapbox/dark-v9', pitch = 45 ) %>%
add_scatterplot(
  data = df
  , lat = "lat"
  , lon = "lng"
  , layer_id = "scatter_layer"
)
```

add_screengrid *Add Screengrid*

Description

The Screen Grid Layer takes in an array of latitude and longitude coordinated points, aggregates them into histogram bins and renders as a grid

Usage

```
add_screengrid(map, data = get_map_data(map), lon = NULL, lat = NULL,
  polyline = NULL, weight = NULL, colour_range = viridisLite::viridis(6),
  opacity = 0.8, cell_size = 50, layer_id, digits = 6)
```

Arguments

map	a mapdeck map object
data	data to be used in the layer
lon	column containing longitude values
lat	column containing latitude values
polyline	column of data containing the polylines
weight	the weight of each value
colour_range	vector of 6 hex colours
opacity	opacity of cells. Value between 0 and 1
cell_size	size of grid squares in pixels
layer_id	single value specifying an id for the layer. Use this value to distinguish between shape layers of the same type
digits	integer. Use this parameter to specify how many digits (decimal places) should be used for the latitude / longitude coordinates.

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

df <- read.csv(paste0(
'https://raw.githubusercontent.com/uber-common/deck.gl-data/master/',
'examples/3d-heatmap/heatmap-data.csv'
))

df$weight <- sample(1:10, size = nrow(df), replace = T)

mapdeck( token = key, style = mapdeck_style('dark'), pitch = 45 ) %>%
```

```
add_screengrid(
  data = df
  , lat = "lat"
  , lon = "lng"
  , weight = "weight",
  , layer_id = "screengrid_layer"
  , cell_size = 10
  , opacity = 0.3
)
```

---

add_text *Add Text*

Description

The Text Layer takes in coordinate points and renders them as circles with a certain radius.

Usage

```
add_text(map, data = get_map_data(map), text, lon = NULL, lat = NULL,
  polyline = NULL, fill_colour = NULL, fill_opacity = NULL, size = NULL,
  angle = NULL, anchor = NULL, alignment_baseline = NULL,
  tooltip = NULL, layer_id, digits = 6, palette = viridisLite::viridis)
```

Arguments

map	a mapdeck map object
data	data to be used in the layer
text	column of data containing the text
lon	column containing longitude values
lat	column containing latitude values
polyline	column of data containing the polylines
fill_colour	column of data or hex colour for the fill colour
fill_opacity	value between 1 and 255. Either a string specifying the column of data containing the fill opacity of each shape, or a value between 1 and 255 to be applied to all the shapes
size	column of data containing the size of the text
angle	column of data containging the angle of the text
anchor	column of data containing the anchor of the text. One of 'start', 'middle' or 'end'
alignment_baseline	column of data containing the alignment. One of 'top', 'center' or 'bottom'
tooltip	variable of data containing text or HTML to render as a tooltip

layer_id	single value specifying an id for the layer. Use this value to distinguish between shape layers of the same type
digits	integer. Use this parameter to specify how many digits (decimal places) should be used for the latitude / longitude coordinates.
palette	a function which generates hex colours

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

mapdeck(
  token = key,
  style = mapdeck_style('dark')
) %>%
  add_text(
    data = capitals
    , lon = 'lon'
    , lat = 'lat'
    , fill_colour = 'country'
    , text = 'capital'
    , layer_id = 'text'
  )
```

---

capitals *Capital cities for each country*

Description

A data set containing the coordinates of 200 capitical cities in the world

Usage

```
capitals
```

Format

A data frame with 200 observations and 4 variables

country country name

capital capital name

lat latitude of capital

lon longitude of capital

clear_tokens *Clear tokens*

Description

Clears the access tokens

Usage

```
clear_tokens()
```

geojson *Geojson*

Description

A GeoJSON object of polygons, lines and points in Melbourne

Usage

```
geojson
```

Format

a 'json' object

light_settings *Light Settings*

Description

List object containg light settings.

Details

Available in add_geojson, add_pointcloud and add_polygon

- numberOfLights - the number of lights. Maximum of 5
- lightsPosition - vector of x, y, z coordinates. Must be 3x the nubmer of lights
- ambientRatio - the ambient ratio of the lights

Examples

```
light <- list(
  lightsPosition = c(-150, 75, 0)
  , numberOfLights = 1
  , ambientRatio = 0.2
)
```

---

mapdeck *mapdeck*

Description

mapdeck

Usage

```
mapdeck(token = get_access_token(api = "mapbox"), data = NULL,
  width = NULL, height = NULL, padding = 0,
  style = "mapbox://styles/mapbox/streets-v9", pitch = 0, zoom = 0,
  location = c(0, 0))
```

Arguments

token	Mapbox Acess token. Use `set_token()` to use a global token
data	data to be used on the map
width	the width of the map
height	the height of the map
padding	the padding of the map
style	the style of the map
pitch	the pitch angle of the map
zoom	zoom level of the map
location	vector of lon and lat coordinates (in that order)

mapdeck-shiny *Shiny bindings for mapdeck*

Description

Output and render functions for using mapdeck within Shiny applications and interactive Rmd documents.

Usage

```
mapdeckOutput(outputId, width = "100%", height = "400px")

renderMapdeck(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a mapdeck
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

mapdeck_dispatch *mapdeck dispatch*

Description

Extension points for plugins

Usage

```
mapdeck_dispatch(map, funcName, mapdeck = stop(paste(funcName,
  "requires a map update object")), mapdeck_update = stop(paste(funcName,
  "does not support map update objects")))

invoke_method(map, method, ...)
```

Arguments

map	a map object, as returned from mapdeck
funcName	the name of the function that the user called that caused this mapdeck_dispatch call; for error message purposes
mapdeck	an action to be performed if the map is from mapdeck
mapdeck_update	an action to be performed if the map is from mapdeck_update
method	the name of the JavaScript method to invoke
...	unnamed arguments to be passed to the JavaScript method

Value

mapdeck_dispatch returns the value of mapdeck or or an error. invokeMethod returns the map object that was passed in, possibly modified.

---

mapdeck_style *Mapdeck Style*

Description

Various styles available to all Mapbox accounts using a valid access token

Usage

```
mapdeck_style(style = c("dark", "light", "outdoors", "streets", "satellite",
  "satellite-streets"))
```

Arguments

style	one of streets, outdoors, light, dark, satellite, satellite-streets

Examples

```
## You need a valid access token from Mapbox
key <- 'abc'

## set a map style
mapdeck(token = key, style = mapdeck_style("dark"))
```

mapdeck_tokens *Mapdeck_tokens*

Description

Retrieves the mapdeck token that has been set

Usage

```
mapdeck_tokens()
```

mapdeck_update *Mapdeck update*

Description

Update a Mapdeck map in a shiny app. Use this function whenever the map needs to respond to reactive content.

Usage

```
mapdeck_update(map_id, session = shiny::getDefaultReactiveDomain(),
  data = NULL, deferUntilFlush = TRUE)
```

Arguments

map_id	string containing the output ID of the map in a shiny application.
session	the Shiny session object to which the map belongs; usually the default value will suffice.
data	data to be used in the map. See the details section for mapdeck.
deferUntilFlush	indicates whether actions performed against this instance should be carried out right away, or whether they should be held until after the next time all of the outputs are updated; defaults to TRUE.

mapdeck_view *Mapdeck view*

Description

Changes the view of the of the map

Usage

```
mapdeck_view(map, location, zoom = 6, duration = 0,
  transition = c("linear", "fly"))
```

Arguments

map	a mapdeck map object
location	vector of lon and lat coordinates (in that order)
zoom	zoom level of the map
duration	time in milliseconds of the transition
transition	type of transition

melbourne *Polygons in and around Melbourne*

Description

A data set containing statistical area 2 regions of central (and surrounds) Melbourne.

Usage

```
melbourne
```

Format

An sfencoded and data frame object with 41 observations and 8 variables. See library googlePolylines for information on sfencoded objects

roads *Roads in central Melbourne*

Description

A simple feature `sf` object of roads in central Melbourne

Usage

```
roads
```

Format

An sf and data frame object with 18286 observations and 16 variables

Details

Obtained from www.data.gov.au and distributed under the Creative Commons 4 License https://creativecommons.org/licenses/by/4.0/

set_token *Set Token*

Description

Sets an access token so it's available for all mapdeck calls. See details

Usage

```
set_token(token)
```

Arguments

`token`	Mapbox access token

Details

Use `set_token` to make access tokens available for all the `mapdeck()` calls in a session so you don't have to keep specifying the `token` argument each time

`%>%` *Pipe*

Description

Uses the pipe operator (%>%) to chain statements. Useful for adding layers to a `google_map`

Arguments

`lhs, rhs`	A google map and a layer to add to it

Examples

```
key <- "your_api_key"
mapdeck_map(key = key) %>%
add_scatterplot(
  data = capitals
  , lat = "lat"
  , lon = "lon"
  , radius = 100000
  , fill_colour = "country"
  , layer_id = "scatter_layer"
)
```

Index