

Package ‘mapgl’

August 17, 2024

Title Interactive Maps with 'Mapbox GL JS' and 'MapLibre GL JS' in R

Version 0.1.2

Date 2024-08-17

Description Provides an interface to the 'Mapbox GL JS' (<<https://docs.mapbox.com/mapbox-gl-js/guides/>>) and the 'MapLibre GL JS' (<<https://maplibre.org/maplibre-gl-js/docs/>>) interactive mapping libraries to help users create custom interactive maps in R. Users can create interactive globe visualizations; layer 'sf' objects to create filled maps, circle maps, 'heatmaps', and three-dimensional graphics; and customize map styles and views. The package also includes utilities to use 'Mapbox' and 'MapLibre' maps in 'Shiny' web applications.

URL <https://walker-data.com/mapgl/>

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Imports htmlwidgets, geojsonsf, sf, rlang, htmltools, grDevices, base64enc, terra, classInt

Suggests shiny, mapboxapi, usethis

NeedsCompilation no

Author Kyle Walker [aut, cre]

Maintainer Kyle Walker <kyle@walker-data.com>

Repository CRAN

Date/Publication 2024-08-17 15:10:02 UTC

Contents

add_categorical_legend	3
add_circle_layer	4
add_continuous_legend	7
add_draw_control	8

add_fill_extrusion_layer	8
add_fill_layer	11
add_fullscreen_control	13
add_heatmap_layer	14
add_image_source	15
add_layer	16
add_layers_control	18
add_legend	19
add_line_layer	20
add_markers	22
add_navigation_control	24
add_raster_dem_source	25
add_raster_layer	25
add_raster_source	27
add_scale_control	28
add_source	29
add_symbol_layer	29
add_vector_source	35
add_video_source	36
carto_style	36
clear_controls	37
clear_layer	37
clear_legend	38
clear_markers	38
compare	39
ease_to	40
fit_bounds	40
fly_to	41
get_column	41
get_drawn_features	42
interpolate	43
jump_to	44
mapboxgl	44
mapboxglOutput	45
mapboxgl_proxy	46
mapbox_style	46
maplibre	47
maplibreOutput	48
maplibre_proxy	48
maptiler_style	49
match_expr	49
renderMapboxgl	50
renderMaplibre	50
set_config_property	51
set_filter	51
set_fog	52
set_layout_property	52
set_paint_property	53

set_style	54
set_terrain	55
set_view	55
step_expr	56

Index**57**

add_categorical_legend

*Add a categorical legend to a Mapbox GL map***Description**

This function adds a categorical legend to a Mapbox GL map. It supports customizable colors, sizes, and shapes for legend items.

Usage

```
add_categorical_legend(
  map,
  legend_title,
  values,
  colors,
  circular_patches = FALSE,
  position = "top-left",
  unique_id = NULL,
  sizes = NULL
)
```

Arguments

map	A map object created by the mapboxgl function.
legend_title	The title of the legend.
values	A vector of categories or values to be displayed in the legend.
colors	The corresponding colors for the values. Can be a vector of colors or a single color.
circular_patches	Logical, whether to use circular patches in the legend. Default is FALSE.
position	The position of the legend on the map. One of "top-left", "bottom-left", "top-right", "bottom-right". Default is "top-left".
unique_id	A unique ID for the legend container. If NULL, a random ID will be generated.
sizes	An optional numeric vector of sizes for the legend patches, or a single numeric value. If provided as a vector, it should have the same length as values. If circular_patches is FALSE (for square patches), sizes represent the width and height of the patch in pixels. If circular_patches is TRUE, sizes represent the radius of the circle.

Value

The updated map object with the legend added.

Examples

```
## Not run:
library(mapboxgl)
map <- mapboxgl(
  center = c(-96, 37.8),
  zoom = 3
)
map %>% add_categorical_legend(
  legend_title = "Population",
  values = c("Low", "Medium", "High"),
  colors = c("#FED976", "#FEB24C", "#FD8D3C"),
  circular_patches = TRUE,
  sizes = c(10, 15, 20)
)

## End(Not run)
```

add_circle_layer	<i>Add a circle layer to a Mapbox GL map</i>
------------------	--

Description

Add a circle layer to a Mapbox GL map

Usage

```
add_circle_layer(
  map,
  id,
  source,
  source_layer = NULL,
  circle_blur = NULL,
  circle_color = NULL,
  circle_opacity = NULL,
  circle_radius = NULL,
  circle_sort_key = NULL,
  circle_stroke_color = NULL,
  circle_stroke_opacity = NULL,
  circle_stroke_width = NULL,
  circle_translate = NULL,
  circle_translate_anchor = "map",
  visibility = "visible",
  slot = NULL,
  min_zoom = NULL,
```

```

    max_zoom = NULL,
    popup = NULL,
    tooltip = NULL,
    hover_options = NULL,
    before_id = NULL
)

```

Arguments

map	A map object created by the <code>mapboxgl</code> function.
id	A unique ID for the layer.
source	The ID of the source, alternatively an <code>sf</code> object (which will be converted to a GeoJSON source) or a named list that specifies <code>type</code> and <code>url</code> for a remote source.
source_layer	The source layer (for vector sources).
circle_blur	Amount to blur the circle.
circle_color	The color of the circle.
circle_opacity	The opacity at which the circle will be drawn.
circle_radius	Circle radius.
circle_sort_key	Sorts features in ascending order based on this value.
circle_stroke_color	The color of the circle's stroke.
circle_stroke_opacity	The opacity of the circle's stroke.
circle_stroke_width	The width of the circle's stroke.
circle_translate	The geometry's offset. Values are <code>c(x, y)</code> where negatives indicate left and up.
circle_translate_anchor	Controls the frame of reference for <code>circle-translate</code> .
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).

Value

The modified map object with the new circle layer added.

Examples

```
## Not run:
library(mapgl)
library(sf)
library(dplyr)

# Set seed for reproducibility
set.seed(1234)

# Define the bounding box for Washington DC (approximately)
bbox <- st_bbox(c(
  xmin = -77.119759,
  ymin = 38.791645,
  xmax = -76.909393,
  ymax = 38.995548
),
crs = st_crs(4326))

# Generate 30 random points within the bounding box
random_points <- st_as_sf(
  data.frame(
    id = 1:30,
    lon = runif(30, bbox["xmin"], bbox["xmax"]),
    lat = runif(30, bbox["ymin"], bbox["ymax"])
  ),
  coords = c("lon", "lat"),
  crs = 4326
)

# Assign random categories
categories <- c('music', 'bar', 'theatre', 'bicycle')
random_points <- random_points %>%
  mutate(category = sample(categories, n(), replace = TRUE))

# Map with circle layer
mapboxgl(style = mapbox_style("light")) %>%
  fit_bounds(random_points, animate = FALSE) %>%
  add_circle_layer(
    id = "poi-layer",
    source = random_points,
    circle_color = match_expr(
      "category",
      values = c("music", "bar", "theatre",
                 "bicycle"),
      stops = c("#1f78b4", "#33a02c",
                "#e31a1c", "#ff7f00")
    ),
    circle_radius = 8,
```

```

        circle_stroke_color = "#ffffff",
        circle_stroke_width = 2,
        circle_opacity = 0.8,
        tooltip = "category",
        hover_options = list(circle_radius = 12,
                             circle_color = "#ffff99")
    ) %>%
    add_categorical_legend(
      legend_title = "Points of Interest",
      values = c("Music", "Bar", "Theatre", "Bicycle"),
      colors = c("#1f78b4", "#33a02c", "#e31a1c", "#ff7f00"),
      circular_patches = TRUE
    )

## End(Not run)

```

add_continuous_legend *Add a continuous legend*

Description

Add a continuous legend

Usage

```

add_continuous_legend(
  map,
  legend_title,
  values,
  colors,
  position = "top-left",
  unique_id = NULL
)

```

Arguments

map	A map object created by the mapboxgl function.
legend_title	The title of the legend.
values	The values being represented on the map (vector of stops).
colors	The colors used to generate the color ramp.
position	The position of the legend on the map (one of "top-left", "bottom-left", "top-right", "bottom-right").
unique_id	A unique ID for the legend container. Defaults to NULL.

Value

The updated map object with the legend added.

add_draw_control *Add a draw control to a map*

Description

Add a draw control to a map

Usage

```
add_draw_control(map, position = "top-left", ...)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
position	A string specifying the position of the draw control. One of "top-right", "top-left", "bottom-right", or "bottom-left".
...	Additional named arguments. See https://github.com/mapbox/mapbox-gl-draw/blob/main/docs/API.md#options for a list of options.

Value

The modified map object with the draw control added.

Examples

```
## Not run:
library(mapgl)

mapboxgl(style = mapbox_style("streets"),
         center = c(-74.50, 40),
         zoom = 9) |>
  add_draw_control(position = "top-left")

## End(Not run)
```

add_fill_extrusion_layer *Add a fill-extrusion layer to a Mapbox GL map*

Description

Add a fill-extrusion layer to a Mapbox GL map

Usage

```

add_fill_extrusion_layer(
  map,
  id,
  source,
  source_layer = NULL,
  fill_extrusion_base = NULL,
  fill_extrusion_color = NULL,
  fill_extrusion_height = NULL,
  fill_extrusion_opacity = NULL,
  fill_extrusion_pattern = NULL,
  fill_extrusion_translate = NULL,
  fill_extrusion_translate_anchor = "map",
  visibility = "visible",
  slot = NULL,
  min_zoom = NULL,
  max_zoom = NULL,
  popup = NULL,
  tooltip = NULL,
  hover_options = NULL,
  before_id = NULL
)

```

Arguments

map	A map object created by the <code>mapboxgl</code> function.
id	A unique ID for the layer.
source	The ID of the source, alternatively an <code>sf</code> object (which will be converted to a GeoJSON source) or a named list that specifies <code>type</code> and <code>url</code> for a remote source.
source_layer	The source layer (for vector sources).
fill_extrusion_base	The base height of the fill extrusion.
fill_extrusion_color	The color of the fill extrusion.
fill_extrusion_height	The height of the fill extrusion.
fill_extrusion_opacity	The opacity of the fill extrusion.
fill_extrusion_pattern	Name of image in sprite to use for drawing image fills.
fill_extrusion_translate	The geometry's offset. Values are <code>c(x, y)</code> where negatives indicate left and up.
fill_extrusion_translate_anchor	Controls the frame of reference for <code>fill-extrusion-translate</code> .
visibility	Whether this layer is displayed.

slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).

Value

The modified map object with the new fill-extrusion layer added.

Examples

```
## Not run:
library(mapgl)

maplibre(
  style = maptiler_style("basic"),
  center = c(-74.0066, 40.7135),
  zoom = 15.5,
  pitch = 45,
  bearing = -17.6
) |>
add_vector_source(
  id = "openmaptiles",
  url = paste0("https://api.maptiler.com/tiles/v3/tiles.json?key=",
    Sys.getenv("MAPTILER_API_KEY"))
) |>
add_fill_extrusion_layer(
  id = "3d-buildings",
  source = 'openmaptiles',
  source_layer = 'building',
  fill_extrusion_color = interpolate(
    column = 'render_height',
    values = c(0, 200, 400),
    stops = c('lightgray', 'royalblue', 'lightblue')
  ),
  fill_extrusion_height = list(
    'interpolate',
    list('linear'),
    list('zoom'),
    15,
    0,
    16,
    list('get', 'render_height')
  )
)
```

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

add_fill_layer	<i>Add a fill layer to a map</i>
----------------	----------------------------------

Description

Add a fill layer to a map

Usage

```
add_fill_layer(  
  map,  
  id,  
  source,  
  source_layer = NULL,  
  fill_antialias = TRUE,  
  fill_color = NULL,  
  fill_emissive_strength = NULL,  
  fill_opacity = NULL,  
  fill_outline_color = NULL,  
  fill_pattern = NULL,  
  fill_sort_key = NULL,  
  fill_translate = NULL,  
  fill_translate_anchor = "map",  
  visibility = "visible",  
  slot = NULL,  
  min_zoom = NULL,  
  max_zoom = NULL,  
  popup = NULL,  
  tooltip = NULL,  
  hover_options = NULL,  
  before_id = NULL  
)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
id	A unique ID for the layer.
source	The ID of the source, alternatively an <code>sf</code> object (which will be converted to a GeoJSON source) or a named list that specifies <code>type</code> and <code>url</code> for a remote source.
source_layer	The source layer (for vector sources).
fill_antialias	Whether or not the fill should be antialiased.

fill_color	The color of the filled part of this layer.
fill_emissive_strength	Controls the intensity of light emitted on the source features.
fill_opacity	The opacity of the entire fill layer.
fill_outline_color	The outline color of the fill.
fill_pattern	Name of image in sprite to use for drawing image fills.
fill_sort_key	Sorts features in ascending order based on this value.
fill_translate	The geometry's offset. Values are c(x, y) where negatives indicate left and up.
fill_translate_anchor	Controls the frame of reference for fill-translate.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).

Value

The modified map object with the new fill layer added.

Examples

```
## Not run:
library(tidycensus)

fl_age <- get_acs(
  geography = "tract",
  variables = "B01002_001",
  state = "FL",
  year = 2022,
  geometry = TRUE
)

mapboxgl() |>
  fit_bounds(fl_age, animate = FALSE) |>
  add_fill_layer(
    id = "fl_tracts",
    source = fl_age,
    fill_color = interpolate(
```

```
      column = "estimate",
      values = c(20, 80),
      stops = c("lightblue", "darkblue"),
      na_color = "lightgrey"
    ),
    fill_opacity = 0.5
  )

## End(Not run)
```

add_fullscreen_control

Add a fullscreen control to a map

Description

Add a fullscreen control to a map

Usage

```
add_fullscreen_control(map, position = "top-right")
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
position	A string specifying the position of the fullscreen control. One of "top-right", "top-left", "bottom-right", or "bottom-left".

Value

The modified map object with the fullscreen control added.

Examples

```
## Not run:
library(mapgl)

maplibre(style = maptiler_style("streets"),
  center = c(11.255, 43.77),
  zoom = 13) |>
  add_fullscreen_control(position = "top-right")

## End(Not run)
```

add_heatmap_layer *Add a heatmap layer to a Mapbox GL map*

Description

Add a heatmap layer to a Mapbox GL map

Usage

```
add_heatmap_layer(  
  map,  
  id,  
  source,  
  source_layer = NULL,  
  heatmap_color = NULL,  
  heatmap_intensity = NULL,  
  heatmap_opacity = NULL,  
  heatmap_radius = NULL,  
  heatmap_weight = NULL,  
  visibility = "visible",  
  slot = NULL,  
  min_zoom = NULL,  
  max_zoom = NULL,  
  before_id = NULL  
)
```

Arguments

map	A map object created by the <code>mapboxgl</code> function.
id	A unique ID for the layer.
source	The ID of the source, alternatively an <code>sf</code> object (which will be converted to a GeoJSON source) or a named list that specifies <code>type</code> and <code>url</code> for a remote source.
source_layer	The source layer (for vector sources).
heatmap_color	The color of the heatmap points.
heatmap_intensity	The intensity of the heatmap points.
heatmap_opacity	The opacity of the heatmap layer.
heatmap_radius	The radius of influence of each individual heatmap point.
heatmap_weight	The weight of each individual heatmap point.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.

max_zoom	The maximum zoom level for the layer.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).

Value

The modified map object with the new heatmap layer added.

Examples

```
## Not run:
library(mapgl)

mapboxgl(style = mapbox_style("dark"),
  center = c(-120, 50),
  zoom = 2) |>
add_heatmap_layer(
  id = "earthquakes-heat",
  source = list(
    type = "geojson",
    data = "https://docs.mapbox.com/mapbox-gl-js/assets/earthquakes.geojson"
  ),
  heatmap_weight = interpolate(
    column = "mag",
    values = c(0, 6),
    stops = c(0, 1)
  ),
  heatmap_intensity = interpolate(
    property = "zoom",
    values = c(0, 9),
    stops = c(1, 3)
  ),
  heatmap_color = interpolate(
    property = "heatmap-density",
    values = seq(0, 1, 0.2),
    stops = c('rgba(33,102,172,0)', 'rgb(103,169,207)',
              'rgb(209,229,240)', 'rgb(253,219,199)',
              'rgb(239,138,98)', 'rgb(178,24,43)')
  ),
  heatmap_opacity = 0.7
)

## End(Not run)
```

add_image_source	<i>Add an image source to a Mapbox GL or Maplibre GL map</i>
------------------	--

Description

Add an image source to a Mapbox GL or Maplibre GL map

Usage

```
add_image_source(
  map,
  id,
  url = NULL,
  data = NULL,
  coordinates = NULL,
  colors = NULL
)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> function.
id	A unique ID for the source.
url	A URL pointing to the image source.
data	A <code>SpatRaster</code> object from the <code>terra</code> package or a <code>RasterLayer</code> object.
coordinates	A list of coordinates specifying the image corners in clockwise order: top left, top right, bottom right, bottom left. For <code>SpatRaster</code> or <code>RasterLayer</code> objects, this will be extracted for you.
colors	A vector of colors to use for the raster image.

Value

The modified map object with the new source added.

add_layer	<i>Add a layer to a map from a source</i>
-----------	---

Description

In many cases, you will use `add_layer()` internal to other layer-specific functions in `mapgl`. Advanced users will want to use `add_layer()` for more fine-grained control over the appearance of their layers.

Usage

```
add_layer(
  map,
  id,
  type = "fill",
  source,
  source_layer = NULL,
  paint = list(),
  layout = list(),
  slot = NULL,
```



```

    min_zoom = NULL,
    max_zoom = NULL,
    popup = NULL,
    tooltip = NULL,
    hover_options = NULL,
    before_id = NULL
  )

```

Arguments

map	A map object created by the <code>mapboxgl()</code> or <code>maplibre()</code> functions.
id	A unique ID for the layer.
type	The type of the layer (e.g., "fill", "line", "circle").
source	The ID of the source, alternatively an <code>sf</code> object (which will be converted to a GeoJSON source) or a named list that specifies <code>type</code> and <code>url</code> for a remote source.
source_layer	The source layer (for vector sources).
paint	A list of paint properties for the layer.
layout	A list of layout properties for the layer.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).

Value

The modified map object with the new layer added.

Examples

```

## Not run:
# Load necessary libraries
library(mapgl)
library(tigris)

# Load geojson data for North Carolina tracts
nc_tracts <- tracts(state = "NC", cb = TRUE)

# Create a Mapbox GL map
map <- mapboxgl(

```

```

    style = mapbox_style("light"),
    center = c(-79.0193, 35.7596),
    zoom = 7
  )

  # Add a source and fill layer for North Carolina tracts
  map %>%
    add_source(
      id = "nc-tracts",
      data = nc_tracts
    ) %>%
    add_layer(
      id = "nc-layer",
      type = "fill",
      source = "nc-tracts",
      paint = list(
        "fill-color" = "#888888",
        "fill-opacity" = 0.4
      )
    )
  )

## End(Not run)

```

add_layers_control *Add a layers control to the map*

Description

Add a layers control to the map

Usage

```

add_layers_control(
  map,
  position = "top-left",
  layers = NULL,
  collapsible = FALSE
)

```

Arguments

map	A map object.
position	The position of the control on the map (one of "top-left", "top-right", "bottom-left", "bottom-right").
layers	A vector of layer IDs to be included in the control. If NULL, all layers will be included.
collapsible	Whether the control should be collapsible.

Value

The modified map object with the layers control added.

Examples

```
## Not run:
library(tigris)
options(tigris_use_cache = TRUE)

rds <- roads("TX", "Tarrant")
tr <- tracts("TX", "Tarrant", cb = TRUE)

maplibre() |>
  fit_bounds(rds) |>
  add_fill_layer(
    id = "Census tracts",
    source = tr,
    fill_color = "purple",
    fill_opacity = 0.6
  ) |>
  add_line_layer(
    "Local roads",
    source = rds,
    line_color = "pink"
  ) |>
  add_layers_control(collapsible = TRUE)

## End(Not run)
```

add_legend

Add a legend to a Mapbox GL map

Description

Add a legend to a Mapbox GL map

Usage

```
add_legend(
  map,
  legend_title,
  values,
  colors,
  type = c("continuous", "categorical"),
  circular_patches = FALSE,
  position = "top-left",
  sizes = NULL
)
```

Arguments

map	A map object created by the mapboxgl function.
legend_title	The title of the legend.
values	The values being represented on the map (either a vector of categories or a vector of stops).
colors	The corresponding colors for the values (either a vector of colors, a single color, or an interpolate function).
type	One of "continuous" or "categorical".
circular_patches	Logical, whether to use circular patches in the legend (only for categorical legends).
position	The position of the legend on the map (one of "top-left", "bottom-left", "top-right", "bottom-right").
sizes	An optional numeric vector of sizes for the legend patches, or a single numeric value (only for categorical legends).

Value

The updated map object with the legend added.

add_line_layer	<i>Add a line layer to a map</i>
----------------	----------------------------------

Description

Add a line layer to a map

Usage

```
add_line_layer(
  map,
  id,
  source,
  source_layer = NULL,
  line_blur = NULL,
  line_color = NULL,
  line_dasharray = NULL,
  line_gap_width = NULL,
  line_offset = NULL,
  line_opacity = NULL,
  line_pattern = NULL,
  line_sort_key = NULL,
  line_translate = NULL,
  line_translate_anchor = "map",
```

```

    line_width = NULL,
    visibility = "visible",
    slot = NULL,
    min_zoom = NULL,
    max_zoom = NULL,
    popup = NULL,
    tooltip = NULL,
    hover_options = NULL,
    before_id = NULL
)

```

Arguments

map	A map object created by the mapboxgl or maplibre functions.
id	A unique ID for the layer.
source	The ID of the source, alternatively an sf object (which will be converted to a GeoJSON source) or a named list that specifies type and url for a remote source.
source_layer	The source layer (for vector sources).
line_blur	Amount to blur the line.
line_color	The color with which the line will be drawn.
line_dasharray	Specifies the lengths of the alternating dashes and gaps that form the dash pattern.
line_gap_width	The width of the gap between a dashed line's individual dashes.
line_offset	The line's offset.
line_opacity	The opacity at which the line will be drawn.
line_pattern	Name of image in sprite to use for drawing image fills.
line_sort_key	Sorts features in ascending order based on this value.
line_translate	The geometry's offset. Values are c(x, y) where negatives indicate left and up.
line_translate_anchor	Controls the frame of reference for line-translate.
line_width	Stroke thickness.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order. Only available when using the Mapbox Standard style.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).

Value

The modified map object with the new line layer added.

Examples

```
## Not run:
library(mapgl)
library(tigris)

loving_roads <- roads("TX", "Loving")

maplibre(style = maptiler_style("backdrop")) |>
  fit_bounds(loving_roads) |>
  add_line_layer(
    id = "tracks",
    source = loving_roads,
    line_color = "navy",
    line_opacity = 0.7
  )

## End(Not run)
```

 add_markers

Add markers to a Mapbox GL or Maplibre GL map

Description

Add markers to a Mapbox GL or Maplibre GL map

Usage

```
add_markers(
  map,
  data,
  color = "red",
  rotation = 0,
  popup = NULL,
  marker_id = NULL,
  draggable = FALSE,
  ...
)
```

Arguments

map	A map object created by the mapboxgl or maplibre functions.
data	A length-2 numeric vector of coordinates, a list of length-2 numeric vectors, or an sf POINT object.
color	The color of the marker (default is "red").

rotation	The rotation of the marker (default is 0).
popup	A column name for popups (if data is an sf object) or a string for a single popup (if data is a numeric vector or list of vectors).
marker_id	A unique ID for the marker. For lists, names will be inherited from the list names. For sf objects, this should be a column name.
draggable	A boolean indicating if the marker should be draggable (default is FALSE).
...	Additional options passed to the marker.

Value

The modified map object with the markers added.

Examples

```
## Not run:
library(mapgl)
library(sf)

# Create a map object
map <- mapboxgl(
  style = mapbox_style("streets"),
  center = c(-74.006, 40.7128),
  zoom = 10
)

# Add a single draggable marker with an ID
map <- add_markers(
  map,
  c(-74.006, 40.7128),
  color = "blue",
  rotation = 45,
  popup = "A marker",
  draggable = TRUE,
  marker_id = "marker1"
)

# Add multiple markers from a named list of coordinates
coords_list <- list(marker2 = c(-74.006, 40.7128),
                    marker3 = c(-73.935242, 40.730610))
map <- add_markers(
  map,
  coords_list,
  color = "green",
  popup = "Multiple markers",
  draggable = TRUE
)

# Create an sf POINT object
points_sf <- st_as_sf(data.frame(
  id = c("marker4", "marker5"),
  lon = c(-74.006, -73.935242),
```

```
    lat = c(40.7128, 40.730610)
  ), coords = c("lon", "lat"), crs = 4326)
points_sf$popup <- c("Point 1", "Point 2")

# Add multiple markers from an sf object with IDs from a column
map <- add_markers(
  map,
  points_sf,
  color = "red",
  popup = "popup",
  draggable = TRUE,
  marker_id = "id"
)

## End(Not run)
```

add_navigation_control

Add a navigation control to a map

Description

Add a navigation control to a map

Usage

```
add_navigation_control(
  map,
  show_compass = TRUE,
  show_zoom = TRUE,
  visualize_pitch = FALSE,
  position = "top-right"
)
```

Arguments

map	A map object created by the mapboxgl or maplibre functions.
show_compass	Whether to show the compass button.
show_zoom	Whether to show the zoom-in and zoom-out buttons.
visualize_pitch	Whether to visualize the pitch by rotating the X-axis of the compass.
position	The position on the map where the control will be added. Possible values are "top-left", "top-right", "bottom-left", and "bottom-right".

Value

The updated map object with the navigation control added.

Examples

```
## Not run:
library(mapgl)

mapboxgl() |>
  add_navigation_control(visualize_pitch = TRUE)

## End(Not run)
```

add_raster_dem_source *Add a raster DEM source to a Mapbox GL or Maplibre GL map*

Description

Add a raster DEM source to a Mapbox GL or Maplibre GL map

Usage

```
add_raster_dem_source(map, id, url, tileSize = 512, maxzoom = NULL)
```

Arguments

map	A map object created by the mapboxgl or maplibre function.
id	A unique ID for the source.
url	A URL pointing to the raster DEM source.
tileSize	The size of the raster tiles.
maxzoom	The maximum zoom level for the raster tiles.

Value

The modified map object with the new source added.

add_raster_layer *Add a raster layer to a Mapbox GL map*

Description

Add a raster layer to a Mapbox GL map

Usage

```

add_raster_layer(
  map,
  id,
  source,
  source_layer = NULL,
  raster_brightness_max = NULL,
  raster_brightness_min = NULL,
  raster_contrast = NULL,
  raster_fade_duration = NULL,
  raster_hue_rotate = NULL,
  raster_opacity = NULL,
  raster_resampling = NULL,
  raster_saturation = NULL,
  visibility = "visible",
  slot = NULL,
  min_zoom = NULL,
  max_zoom = NULL,
  before_id = NULL
)

```

Arguments

map	A map object created by the <code>mapboxgl</code> function.
id	A unique ID for the layer.
source	The ID of the source.
source_layer	The source layer (for vector sources).
raster_brightness_max	The maximum brightness of the image.
raster_brightness_min	The minimum brightness of the image.
raster_contrast	Increase or reduce the brightness of the image.
raster_fade_duration	The duration of the fade-in/fade-out effect.
raster_hue_rotate	Rotates hues around the color wheel.
raster_opacity	The opacity at which the raster will be drawn.
raster_resampling	The resampling/interpolation method to use for overscaling.
raster_saturation	Increase or reduce the saturation of the image.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.

max_zoom	The maximum zoom level for the layer.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).

Value

The modified map object with the new raster layer added.

Examples

```
## Not run:
mapboxgl(style = mapbox_style("dark"),
  zoom = 5,
  center = c(-75.789, 41.874)) |>
  add_image_source(
    id = "radar",
    url = "https://docs.mapbox.com/mapbox-gl-js/assets/radar.gif",
    coordinates = list(
      c(-80.425, 46.437),
      c(-71.516, 46.437),
      c(-71.516, 37.936),
      c(-80.425, 37.936)
    )
  ) |>
  add_raster_layer(
    id = 'radar-layer',
    source = 'radar',
    raster_fade_duration = 0
  )

## End(Not run)
```

add_raster_source *Add a raster tile source to a Mapbox GL or Maplibre GL map*

Description

Add a raster tile source to a Mapbox GL or Maplibre GL map

Usage

```
add_raster_source(
  map,
  id,
  url = NULL,
  tiles = NULL,
  tileSize = 256,
  maxzoom = 22
)
```

Arguments

map	A map object created by the mapboxgl or maplibre function.
id	A unique ID for the source.
url	A URL pointing to the raster tile source. (optional)
tiles	A vector of tile URLs for the raster source. (optional)
tileSize	The size of the raster tiles.
maxzoom	The maximum zoom level for the raster tiles.

Value

The modified map object with the new source added.

add_scale_control *Add a scale control to a map*

Description

This function adds a scale control to a Mapbox GL or Maplibre GL map.

Usage

```
add_scale_control(
  map,
  position = "bottom-left",
  unit = "metric",
  max_width = 100
)
```

Arguments

map	A map object created by the mapboxgl or maplibre functions.
position	The position of the control. Can be one of "top-left", "top-right", "bottom-left", or "bottom-right". Default is "bottom-left".
unit	The unit of the scale. Can be either "imperial", "metric", or "nautical". Default is "metric".
max_width	The maximum length of the scale control in pixels. Default is 100.

Value

The modified map object with the scale control added.

Examples

```
## Not run:
library(mapgl)

mapboxgl() |>
  add_scale_control(position = "bottom-right", unit = "imperial")

## End(Not run)
```

add_source	<i>Add a GeoJSON or sf source to a Mapbox GL or Maplibre GL map</i>
------------	---

Description

Add a GeoJSON or sf source to a Mapbox GL or Maplibre GL map

Usage

```
add_source(map, id, data, ...)
```

Arguments

map	A map object created by the mapboxgl or maplibre function.
id	A unique ID for the source.
data	An sf object or a URL pointing to a remote GeoJSON file.
...	Additional arguments to be passed to the JavaScript addSource method.

Value

The modified map object with the new source added.

add_symbol_layer	<i>Add a symbol layer to a map</i>
------------------	------------------------------------

Description

Add a symbol layer to a map

Usage

```
add_symbol_layer(  
    map,  
    id,  
    source,  
    source_layer = NULL,  
    icon_allow_overlap = NULL,  
    icon_anchor = NULL,  
    icon_color = NULL,  
    icon_color_brightness_max = NULL,  
    icon_color_brightness_min = NULL,  
    icon_color_contrast = NULL,  
    icon_color_saturation = NULL,  
    icon_emissive_strength = NULL,  
    icon_halo_blur = NULL,  
    icon_halo_color = NULL,  
    icon_halo_width = NULL,  
    icon_ignore_placement = NULL,  
    icon_image = NULL,  
    icon_image_cross_fade = NULL,  
    icon_keep_upright = NULL,  
    icon_offset = NULL,  
    icon_opacity = NULL,  
    icon_optional = NULL,  
    icon_padding = NULL,  
    icon_pitch_alignment = NULL,  
    icon_rotate = NULL,  
    icon_rotation_alignment = NULL,  
    icon_size = NULL,  
    icon_text_fit = NULL,  
    icon_text_fit_padding = NULL,  
    icon_translate = NULL,  
    icon_translate_anchor = NULL,  
    symbol_avoid_edges = NULL,  
    symbol_placement = NULL,  
    symbol_sort_key = NULL,  
    symbol_spacing = NULL,  
    symbol_z_elevate = NULL,  
    symbol_z_order = NULL,  
    text_allow_overlap = NULL,  
    text_anchor = NULL,  
    text_color = NULL,  
    text_emissive_strength = NULL,  
    text_field = NULL,  
    text_font = NULL,  
    text_halo_blur = NULL,  
    text_halo_color = NULL,  
    text_halo_width = NULL,
```

```

text_ignore_placement = NULL,
text_justify = NULL,
text_keep_upright = NULL,
text_letter_spacing = NULL,
text_line_height = NULL,
text_max_angle = NULL,
text_max_width = NULL,
text_offset = NULL,
text_opacity = NULL,
text_optional = NULL,
text_padding = NULL,
text_pitch_alignment = NULL,
text_radial_offset = NULL,
text_rotate = NULL,
text_rotation_alignment = NULL,
text_size = NULL,
text_transform = NULL,
text_translate = NULL,
text_translate_anchor = NULL,
text_variable_anchor = NULL,
text_writing_mode = NULL,
visibility = "visible",
slot = NULL,
min_zoom = NULL,
max_zoom = NULL,
popup = NULL,
tooltip = NULL,
hover_options = NULL,
before_id = NULL
)

```

Arguments

map	A map object created by the mapboxgl or maplibre functions.
id	A unique ID for the layer.
source	The ID of the source, alternatively an sf object (which will be converted to a GeoJSON source) or a named list that specifies type and url for a remote source.
source_layer	The source layer (for vector sources).
icon_allow_overlap	If TRUE, the icon will be visible even if it collides with other previously drawn symbols.
icon_anchor	Part of the icon placed closest to the anchor.
icon_color	The color of the icon. This is not supported for many Mapbox icons; read more at https://docs.mapbox.com/help/troubleshooting/using-recolorable-images-in-mapbox-m
icon_color_brightness_max	The maximum brightness of the icon color.

<code>icon_color_brightness_min</code>	The minimum brightness of the icon color.
<code>icon_color_contrast</code>	The contrast of the icon color.
<code>icon_color_saturation</code>	The saturation of the icon color.
<code>icon_emissive_strength</code>	The strength of the icon's emissive color.
<code>icon_halo_blur</code>	The blur applied to the icon's halo.
<code>icon_halo_color</code>	The color of the icon's halo.
<code>icon_halo_width</code>	The width of the icon's halo.
<code>icon_ignore_placement</code>	If TRUE, the icon will be visible even if it collides with other symbols.
<code>icon_image</code>	Name of image in sprite to use for drawing an image background. To use values in a column of your input dataset, use <code>c('get', 'YOUR_ICON_COLUMN_NAME')</code> .
<code>icon_image_cross_fade</code>	The cross-fade parameter for the icon image.
<code>icon_keep_upright</code>	If TRUE, the icon will be kept upright.
<code>icon_offset</code>	Offset distance of icon.
<code>icon_opacity</code>	The opacity at which the icon will be drawn.
<code>icon_optional</code>	If TRUE, the icon will be optional.
<code>icon_padding</code>	Padding around the icon.
<code>icon_pitch_alignment</code>	Alignment of the icon with respect to the pitch of the map.
<code>icon_rotate</code>	Rotates the icon clockwise.
<code>icon_rotation_alignment</code>	Alignment of the icon with respect to the map.
<code>icon_size</code>	The size of the icon.
<code>icon_text_fit</code>	Scales the text to fit the icon.
<code>icon_text_fit_padding</code>	Padding for text fitting the icon.
<code>icon_translate</code>	The offset distance of the icon.
<code>icon_translate_anchor</code>	Controls the frame of reference for icon-translate.
<code>symbol_avoid_edges</code>	If TRUE, the symbol will be avoided when near the edges.
<code>symbol_placement</code>	Placement of the symbol on the map.
<code>symbol_sort_key</code>	Sorts features in ascending order based on this value.

symbol_spacing Spacing between symbols.

symbol_z_elevate Elevates the symbol z-axis.

symbol_z_order Orders the symbol z-axis.

text_allow_overlap If TRUE, the text will be visible even if it collides with other previously drawn symbols.

text_anchor Part of the text placed closest to the anchor.

text_color The color of the text.

text_emissive_strength The strength of the text's emissive color.

text_field Value to use for a text label.

text_font Font stack to use for displaying text.

text_halo_blur The blur applied to the text's halo.

text_halo_color The color of the text's halo.

text_halo_width The width of the text's halo.

text_ignore_placement If TRUE, the text will be visible even if it collides with other symbols.

text_justify The justification of the text.

text_keep_upright If TRUE, the text will be kept upright.

text_letter_spacing Spacing between text letters.

text_line_height Height of the text lines.

text_max_angle Maximum angle of the text.

text_max_width Maximum width of the text.

text_offset Offset distance of text.

text_opacity The opacity at which the text will be drawn.

text_optional If TRUE, the text will be optional.

text_padding Padding around the text.

text_pitch_alignment Alignment of the text with respect to the pitch of the map.

text_radial_offset Radial offset of the text.

text_rotate Rotates the text clockwise.

text_rotation_alignment Alignment of the text with respect to the map.

text_size The size of the text.

text_transform Transform applied to the text.

text_translate	The offset distance of the text.
text_translate_anchor	Controls the frame of reference for text-translate.
text_variable_anchor	Variable anchor for the text.
text_writing_mode	Writing mode for the text.
visibility	Whether this layer is displayed.
slot	An optional slot for layer order.
min_zoom	The minimum zoom level for the layer.
max_zoom	The maximum zoom level for the layer.
popup	A column name containing information to display in a popup on click. Columns containing HTML will be parsed.
tooltip	A column name containing information to display in a tooltip on hover. Columns containing HTML will be parsed.
hover_options	A named list of options for highlighting features in the layer on hover. Not all elements of SVG icons can be styled.
before_id	The name of the layer that this layer appears "before", allowing you to insert layers below other layers in your basemap (e.g. labels).

Value

The modified map object with the new symbol layer added.

Examples

```
## Not run:
library(mapgl)
library(sf)
library(dplyr)

# Set seed for reproducibility
set.seed(1234)

# Define the bounding box for Washington DC (approximately)
bbox <- st_bbox(c(
  xmin = -77.119759,
  ymin = 38.791645,
  xmax = -76.909393,
  ymax = 38.995548
),
crs = st_crs(4326))

# Generate 30 random points within the bounding box
random_points <- st_as_sf(
  data.frame(
    id = 1:30,
    lon = runif(30, bbox["xmin"], bbox["xmax"]),
```

```

    lat = runif(30, bbox["ymin"], bbox["ymax"])
  ),
  coords = c("lon", "lat"),
  crs = 4326
)

# Assign random icons
icons <- c('music', 'bar', 'theatre', 'bicycle')
random_points <- random_points |>
  mutate(icon = sample(icons, n(), replace = TRUE))

# Map with icons
mapboxgl(style = mapbox_style("light")) |>
  fit_bounds(random_points, animate = FALSE) |>
  add_symbol_layer(
    id = "points-of-interest",
    source = random_points,
    icon_image = c("get", "icon"),
    icon_allow_overlap = TRUE,
    tooltip = "icon"
  )

## End(Not run)

```

add_vector_source	<i>Add a vector tile source to a Mapbox GL or Maplibre GL map</i>
-------------------	---

Description

Add a vector tile source to a Mapbox GL or Maplibre GL map

Usage

```
add_vector_source(map, id, url)
```

Arguments

map	A map object created by the mapboxgl or maplibre function.
id	A unique ID for the source.
url	A URL pointing to the vector tile source.

Value

The modified map object with the new source added.

`add_video_source` *Add a video source to a Mapbox GL or Maplibre GL map*

Description

Add a video source to a Mapbox GL or Maplibre GL map

Usage

```
add_video_source(map, id, urls, coordinates)
```

Arguments

<code>map</code>	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> function.
<code>id</code>	A unique ID for the source.
<code>urls</code>	A vector of URLs pointing to the video sources.
<code>coordinates</code>	A list of coordinates specifying the video corners in clockwise order: top left, top right, bottom right, bottom left.

Value

The modified map object with the new source added.

`carto_style` *Get CARTO Style URL*

Description

Get CARTO Style URL

Usage

```
carto_style(style_name)
```

Arguments

<code>style_name</code>	The name of the style (e.g., "voyager", "positron", "dark-matter").
-------------------------	---

Value

The style URL corresponding to the given style name.

clear_controls	<i>Clear all controls from a Mapbox GL or Maplibre GL map in a Shiny app</i>
----------------	--

Description

Clear all controls from a Mapbox GL or Maplibre GL map in a Shiny app

Usage

```
clear_controls(map)
```

Arguments

map	A map object created by the mapboxgl or maplibre function.
-----	--

Value

The modified map object with all controls removed.

clear_layer	<i>Clear a layer from a map using a proxy</i>
-------------	---

Description

This function allows a layer to be removed from an existing Mapbox GL map using a proxy object.

Usage

```
clear_layer(proxy, layer_id)
```

Arguments

proxy	A proxy object created by mapboxgl_proxy or maplibre_proxy.
layer_id	The ID of the layer to be removed.

Value

The updated proxy object.

clear_legend	<i>Clear legend from a map in a proxy session</i>
--------------	---

Description

Clear legend from a map in a proxy session

Usage

```
clear_legend(map)
```

Arguments

map	A map object created by the <code>mapboxgl_proxy</code> or <code>maplibre_proxy</code> function.
-----	--

Value

The updated map object with the legend cleared.

clear_markers	<i>Clear markers from a map in a Shiny session</i>
---------------	--

Description

Clear markers from a map in a Shiny session

Usage

```
clear_markers(map)
```

Arguments

map	A map object created by the <code>mapboxgl_proxy</code> or <code>maplibre_proxy</code> function.
-----	--

Value

The modified map object with the markers cleared.

`compare`*Create a Compare slider widget*

Description

This function creates a comparison view between two Mapbox GL or Maplibre GL maps, allowing users to swipe between the two maps to compare different styles or data layers.

Usage

```
compare(  
  map1,  
  map2,  
  width = "100%",  
  height = NULL,  
  elementId = NULL,  
  mousemove = FALSE,  
  orientation = "vertical"  
)
```

Arguments

<code>map1</code>	A <code>mapboxgl</code> or <code>maplibre</code> object representing the first map.
<code>map2</code>	A <code>mapboxgl</code> or <code>maplibre</code> object representing the second map.
<code>width</code>	Width of the map container.
<code>height</code>	Height of the map container.
<code>elementId</code>	An optional string specifying the ID of the container for the comparison. If <code>NULL</code> , a unique ID will be generated.
<code>mousemove</code>	A logical value indicating whether to enable swiping during cursor movement (rather than only when clicked).
<code>orientation</code>	A string specifying the orientation of the swiper, either "horizontal" or "vertical".

Value

A comparison widget.

Examples

```
## Not run:  
library(mapgl)  
  
library(mapgl)  
  
m1 <- mapboxgl(style = mapbox_style("light"))  
  
m2 <- mapboxgl(style = mapbox_style("dark"))
```

```
compare(m1, m2)
```

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

ease_to	<i>Ease to a given view</i>
---------	-----------------------------

Description

Ease to a given view

Usage

```
ease_to(map, center, zoom = NULL, ...)
```

Arguments

map	A map object created by the mapboxgl or maplibre function or a proxy object.
center	A numeric vector of length 2 specifying the target center of the map (longitude, latitude).
zoom	The target zoom level.
...	Additional named arguments for easing to the view.

Value

The updated map object.

fit_bounds	<i>Fit the map to a bounding box</i>
------------	--------------------------------------

Description

Fit the map to a bounding box

Usage

```
fit_bounds(map, bbox, animate = FALSE, ...)
```

Arguments

map	A map object created by the mapboxgl or maplibre function or a proxy object.
bbox	A bounding box specified as a numeric vector of length 4 (minLng, minLat, maxLng, maxLat), or an sf object from which a bounding box will be calculated.
animate	A logical value indicating whether to animate the transition to the new bounds. Defaults to FALSE.
...	Additional named arguments for fitting the bounds.

Value

The updated map object.

fly_to	<i>Fly to a given view</i>
--------	----------------------------

Description

Fly to a given view

Usage

```
fly_to(map, center, zoom = NULL, ...)
```

Arguments

map	A map object created by the mapboxgl or maplibre function or a proxy object.
center	A numeric vector of length 2 specifying the target center of the map (longitude, latitude).
zoom	The target zoom level.
...	Additional named arguments for flying to the view.

Value

The updated map object.

get_column	<i>Get column or property for use in mapping</i>
------------	--

Description

This function returns an expression to get a specified column from a dataset (or a property from a layer).

Usage

```
get_column(column)
```

Arguments

column	The name of the column or property to get.
--------	--

Value

A list representing the expression to get the column.

get_drawn_features *Get drawn features from the map*

Description

Get drawn features from the map

Usage

```
get_drawn_features(map)
```

Arguments

map A map object created by the mapboxgl function, or a mapboxgl proxy.

Value

An sf object containing the drawn features.

Examples

```
## Not run:
# In a Shiny application
library(shiny)
library(mapboxer)

ui <- fluidPage(
  mapboxglOutput("map"),
  actionButton("get_features", "Get Drawn Features"),
  verbatimTextOutput("feature_output")
)

server <- function(input, output, session) {
  output$map <- renderMapboxgl({
    mapboxgl(style = mapbox_style("streets-v12"),
             center = c(-74.50, 40),
             zoom = 9) |>
    add_draw_control()
  })

  observeEvent(input$get_features, {
    drawn_features <- get_drawn_features(mapboxgl_proxy("map"))
    output$feature_output <- renderPrint({
      print(drawn_features)
    })
  })
}

shinyApp(ui, server)
```

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

interpolate	<i>Create an interpolation expression</i>
-------------	---

Description

This function generates an interpolation expression that can be used to style your data.

Usage

```
interpolate(  
  column = NULL,  
  property = NULL,  
  type = "linear",  
  values,  
  stops,  
  na_color = NULL  
)
```

Arguments

column	The name of the column to use for the interpolation. If specified, property should be NULL.
property	The name of the property to use for the interpolation. If specified, column should be NULL.
type	The interpolation type. Can be one of "linear", c("exponential", base) where base specifies the rate at which the output increases, or c("cubic-bezier", x1, y1, x2, y2) where you define a cubic bezier curve with control points.
values	A numeric vector of values at which stops occur.
stops	A vector of corresponding stops (colors, sizes, etc.) for the interpolation.
na_color	The color to use for missing values. Mapbox GL JS defaults to black if this is not supplied.

Value

A list representing the interpolation expression.

Examples

```
interpolate(  
  column = "estimate",  
  type = "linear",  
  values = c(1000, 200000),  
  stops = c("#eff3ff", "#08519c")  
)
```

jump_to	<i>Jump to a given view</i>
---------	-----------------------------

Description

Jump to a given view

Usage

```
jump_to(map, center, zoom = NULL, ...)
```

Arguments

map	A map object created by the mapboxgl or maplibre function or a proxy object.
center	A numeric vector of length 2 specifying the target center of the map (longitude, latitude).
zoom	The target zoom level.
...	Additional named arguments for jumping to the view.

Value

The updated map object.

mapboxgl	<i>Initialize a Mapbox GL Map</i>
----------	-----------------------------------

Description

Initialize a Mapbox GL Map

Usage

```
mapboxgl(
  style = NULL,
  center = c(0, 0),
  zoom = 0,
  bearing = 0,
  pitch = 0,
  projection = "globe",
  parallels = NULL,
  access_token = NULL,
  bounds = NULL,
  width = "100%",
  height = NULL,
  ...
)
```

Arguments

style	The Mapbox style to use.
center	A numeric vector of length 2 specifying the initial center of the map.
zoom	The initial zoom level of the map.
bearing	The initial bearing (rotation) of the map, in degrees.
pitch	The initial pitch (tilt) of the map, in degrees.
projection	The map projection to use (e.g., "mercator", "globe").
parallels	A vector of two numbers representing the standard parallels of the projection. Only available when the projection is "albers" or "lambertConformalConic".
access_token	Your Mapbox access token.
bounds	An sf object or bounding box to fit the map to.
width	The width of the output htmlwidget.
height	The height of the output htmlwidget.
...	Additional named parameters to be passed to the Mapbox GL map.

Value

An HTML widget for a Mapbox map.

Examples

```
## Not run:
mapboxgl(projection = "globe")

## End(Not run)
```

mapboxglOutput	<i>Create a Mapbox GL output element for Shiny</i>
----------------	--

Description

Create a Mapbox GL output element for Shiny

Usage

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

Arguments

outputId	The output variable to read from
width	The width of the element
height	The height of the element

Value

A Mapbox GL output element for use in a Shiny UI

mapboxgl_proxy	<i>Create a proxy object for a Mapbox GL map in Shiny</i>
----------------	---

Description

This function allows updates to be sent to an existing Mapbox GL map in a Shiny application without redrawing the entire map.

Usage

```
mapboxgl_proxy(mapId, session = shiny::getDefaultReactiveDomain())
```

Arguments

mapId	The ID of the map output element.
session	The Shiny session object.

Value

A proxy object for the Mapbox GL map.

mapbox_style	<i>Get Mapbox Style URL</i>
--------------	-----------------------------

Description

Get Mapbox Style URL

Usage

```
mapbox_style(style_name)
```

Arguments

style_name	The name of the style (e.g., "standard", "streets", "outdoors", etc.).
------------	--

Value

The style URL corresponding to the given style name.

maplibre	<i>Initialize a Maplibre GL Map</i>
----------	-------------------------------------

Description

Initialize a Maplibre GL Map

Usage

```
maplibre(  
  style = carto_style("voyager"),  
  center = c(0, 0),  
  zoom = 0,  
  bearing = 0,  
  pitch = 0,  
  bounds = NULL,  
  width = "100%",  
  height = NULL,  
  ...  
)
```

Arguments

style	The style JSON to use.
center	A numeric vector of length 2 specifying the initial center of the map.
zoom	The initial zoom level of the map.
bearing	The initial bearing (rotation) of the map, in degrees.
pitch	The initial pitch (tilt) of the map, in degrees.
bounds	An sf object or bounding box to fit the map to.
width	The width of the output htmlwidget.
height	The height of the output htmlwidget.
...	Additional named parameters to be passed to the Mapbox GL map.

Value

An HTML widget for a Mapbox map.

Examples

```
## Not run:  
maplibre()  
  
## End(Not run)
```

maplibreOutput	<i>Create a Maplibre GL output element for Shiny</i>
----------------	--

Description

Create a Maplibre GL output element for Shiny

Usage

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

Arguments

outputId	The output variable to read from
width	The width of the element
height	The height of the element

Value

A Maplibre GL output element for use in a Shiny UI

maplibre_proxy	<i>Create a proxy object for a Maplibre GL map in Shiny</i>
----------------	---

Description

This function allows updates to be sent to an existing Maplibre GL map in a Shiny application without redrawing the entire map.

Usage

```
maplibre_proxy(mapId, session = shiny::getDefaultReactiveDomain())
```

Arguments

mapId	The ID of the map output element.
session	The Shiny session object.

Value

A proxy object for the Maplibre GL map.

maptiler_style	<i>Get MapTiler Style URL</i>
----------------	-------------------------------

Description

Get MapTiler Style URL

Usage

```
maptiler_style(style_name, api_key = NULL)
```

Arguments

style_name	The name of the style (e.g., "basic", "streets", "toner", etc.).
api_key	Your MapTiler API key (required)

Value

The style URL corresponding to the given style name.

match_expr	<i>Create a match expression</i>
------------	----------------------------------

Description

This function generates a match expression that can be used to style your data.

Usage

```
match_expr(column = NULL, property = NULL, values, stops, default = "#cccccc")
```

Arguments

column	The name of the column to use for the match expression. If specified, property should be NULL.
property	The name of the property to use for the match expression. If specified, column should be NULL.
values	A vector of values to match against.
stops	A vector of corresponding stops (colors, etc.) for the matched values.
default	A default value to use if no matches are found.

Value

A list representing the match expression.

Examples

```

match_expr(
  column = "category",
  values = c("A", "B", "C"),
  stops = c("#ff0000", "#00ff00", "#0000ff"),
  default = "#cccccc"
)

```

renderMapboxgl	<i>Render a Mapbox GL output element in Shiny</i>
----------------	---

Description

Render a Mapbox GL output element in Shiny

Usage

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

Arguments

expr	An expression that generates a Mapbox GL map
env	The environment in which to evaluate expr
quoted	Is expr a quoted expression

Value

A rendered Mapbox GL map for use in a Shiny server

renderMaplibre	<i>Render a Maplibre GL output element in Shiny</i>
----------------	---

Description

Render a Maplibre GL output element in Shiny

Usage

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

Arguments

expr	An expression that generates a Maplibre GL map
env	The environment in which to evaluate expr
quoted	Is expr a quoted expression

Value

A rendered Maplibre GL map for use in a Shiny server

set_config_property *Set a configuration property for a Mapbox GL map*

Description

Set a configuration property for a Mapbox GL map

Usage

```
set_config_property(map, import_id, config_name, value)
```

Arguments

map	A map object created by the mapboxgl function or a proxy object defined with mapboxgl_proxy().
import_id	The name of the imported style to set the config for (e.g., 'basemap').
config_name	The name of the configuration property from the style.
value	The value to set for the configuration property.

Value

The updated map object with the configuration property set.

set_filter *Set a filter on a map layer*

Description

This function sets a filter on a map layer, working with both regular map objects and proxy objects.

Usage

```
set_filter(map, layer_id, filter)
```

Arguments

map	A map object created by the mapboxgl or maplibre function, or a proxy object.
layer_id	The ID of the layer to which the filter will be applied.
filter	The filter expression to apply.

Value

The updated map object.

set_fog *Set fog on a Mapbox GL map*

Description

Set fog on a Mapbox GL map

Usage

```
set_fog(
  map,
  range = NULL,
  color = NULL,
  horizon_blend = NULL,
  high_color = NULL,
  space_color = NULL,
  star_intensity = NULL
)
```

Arguments

map	A map object created by the mapboxgl function or a proxy object.
range	A numeric vector of length 2 defining the minimum and maximum range of the fog.
color	A string specifying the color of the fog.
horizon_blend	A number between 0 and 1 controlling the blending of the fog at the horizon.
high_color	A string specifying the color of the fog at higher elevations.
space_color	A string specifying the color of the fog in space.
star_intensity	A number between 0 and 1 controlling the intensity of the stars in the fog.

Value

The updated map object.

set_layout_property *Set a layout property on a map layer*

Description

Set a layout property on a map layer

Usage

```
set_layout_property(map, layer, name, value)
```

Arguments

map	A map object created by the mapboxgl or maplibre function, or a proxy object.
layer	The ID of the layer to update.
name	The name of the layout property to set.
value	The value to set the property to.

Value

The updated map object.

set_paint_property *Set a paint property on a map layer*

Description

Set a paint property on a map layer

Usage

```
set_paint_property(map, layer, name, value)
```

Arguments

map	A map object created by the mapboxgl or maplibre function, or a proxy object.
layer	The ID of the layer to update.
name	The name of the paint property to set.
value	The value to set the property to.

Value

The updated map object.

set_style	<i>Update the style of a map</i>
-----------	----------------------------------

Description

Update the style of a map

Usage

```
set_style(map, style, config = NULL, diff = TRUE)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> function, or a proxy object.
style	The new style URL to be applied to the map.
config	A named list of options to be passed to the style config.
diff	A boolean that attempts a diff-based update rather than re-drawing the full style. Not available for all styles.

Value

The modified map object.

Examples

```
## Not run:
map <- mapboxgl(
  style = mapbox_style("streets"),
  center = c(-74.006, 40.7128),
  zoom = 10,
  access_token = "your_mapbox_access_token"
)

# Update the map style in a Shiny app
observeEvent(input$change_style, {
  mapboxgl_proxy("map", session) %>%
    set_style(mapbox_style("dark"), config = list(showLabels = FALSE), diff = TRUE)
})

## End(Not run)
```

set_terrain	<i>Set terrain properties on a map</i>
-------------	--

Description

Set terrain properties on a map

Usage

```
set_terrain(map, source, exaggeration = 1)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> functions.
source	The ID of the raster DEM source.
exaggeration	The terrain exaggeration factor.

Value

The modified map object with the terrain settings applied.

Examples

```
## Not run:
map <- mapboxgl(style = "mapbox://styles/mapbox/satellite-streets-v12",
  center = c(-114.26608, 32.7213), zoom = 14, pitch = 80, bearing = 41,
  access_token = "your_token_here")
map <- add_source(map, id = "mapbox-dem", type = "raster-dem",
  url = "mapbox://mapbox.mapbox-terrain-dem-v1",
  tileSize = 512, maxzoom = 14)
map <- set_terrain(map, source = "mapbox-dem", exaggeration = 1.5)

## End(Not run)
```

set_view	<i>Set the map center and zoom level</i>
----------	--

Description

Set the map center and zoom level

Usage

```
set_view(map, center, zoom)
```

Arguments

map	A map object created by the <code>mapboxgl</code> or <code>maplibre</code> function or a proxy object.
center	A numeric vector of length 2 specifying the center of the map (longitude, latitude).
zoom	The zoom level.

Value

The updated map object.

step_expr	<i>Create a step expression</i>
-----------	---------------------------------

Description

This function generates a step expression that can be used in your styles.

Usage

```
step_expr(column = NULL, property = NULL, base, values, stops, na_color = NULL)
```

Arguments

column	The name of the column to use for the step expression. If specified, property should be NULL.
property	The name of the property to use for the step expression. If specified, column should be NULL.
base	The base value to use for the step expression.
values	A numeric vector of values at which steps occur.
stops	A vector of corresponding stops (colors, sizes, etc.) for the steps.
na_color	The color to use for missing values. Mapbox GL JS defaults to black if this is not supplied.

Value

A list representing the step expression.

Examples

```
step_expr(
  column = "value",
  base = "#ffffff",
  values = c(1000, 5000, 10000),
  stops = c("#ff0000", "#00ff00", "#0000ff")
)
```


Index

[add_categorical_legend](#), 3
[add_circle_layer](#), 4
[add_continuous_legend](#), 7
[add_draw_control](#), 8
[add_fill_extrusion_layer](#), 8
[add_fill_layer](#), 11
[add_fullscreen_control](#), 13
[add_heatmap_layer](#), 14
[add_image_source](#), 15
[add_layer](#), 16
[add_layers_control](#), 18
[add_legend](#), 19
[add_line_layer](#), 20
[add_markers](#), 22
[add_navigation_control](#), 24
[add_raster_dem_source](#), 25
[add_raster_layer](#), 25
[add_raster_source](#), 27
[add_scale_control](#), 28
[add_source](#), 29
[add_symbol_layer](#), 29
[add_vector_source](#), 35
[add_video_source](#), 36

[carto_style](#), 36
[clear_controls](#), 37
[clear_layer](#), 37
[clear_legend](#), 38
[clear_markers](#), 38
[compare](#), 39

[ease_to](#), 40

[fit_bounds](#), 40
[fly_to](#), 41

[get_column](#), 41
[get_drawn_features](#), 42

[interpolate](#), 43

[jump_to](#), 44

[mapbox_style](#), 46
[mapboxgl](#), 44
[mapboxgl_proxy](#), 46
[mapboxglOutput](#), 45
[maplibre](#), 47
[maplibre_proxy](#), 48
[maplibreOutput](#), 48
[maptiler_style](#), 49
[match_expr](#), 49

[renderMapboxgl](#), 50
[renderMaplibre](#), 50

[set_config_property](#), 51
[set_filter](#), 51
[set_fog](#), 52
[set_layout_property](#), 52
[set_paint_property](#), 53
[set_style](#), 54
[set_terrain](#), 55
[set_view](#), 55
[step_expr](#), 56