Package ‘mapboxer’

November 4, 2020

Title  An R Interface to ‘Mapbox GL JS’
Version  0.4.0
Date  2020-10-28
Maintainer  Stefan Kuethe <crazycapivara@gmail.com>
Description  Makes ‘Mapbox GL JS’ <https://docs.mapbox.com/mapbox-gl-js/api/>,
an open source JavaScript library that uses WebGL to render interactive maps,
available within R via the ‘htmlwidgets’ package. Visualizations can be used from the R console,
in R Markdown documents and in Shiny apps.
License  MIT + file LICENSE
Depends  R (>= 2.10)
Imports  magrittr, htmlwidgets, htmltools, yamll, purrr, geojsonsf,
         methods
Encoding  UTF-8
LazyData  true
RoxygenNote  6.1.1
Suggests  knitr, rmarkdown, shiny
VignetteBuilder  knitr
URL  https://github.com/crazycapivara/mapboxer
BugReports  https://github.com/crazycapivara/mapboxer/issues
NeedsCompilation  no
Author  Stefan Kuethe [aut, cre]
Repository  CRAN
Date/Publication  2020-11-04 16:10:02 UTC

R topics documented:

    add_circle_layer ..................................................  2
    add_control ..........................................................  4
    add_draw_control ......................................................  5
**add_circle_layer**

Add a circle layer to the map

**Description**
Add a circle layer to the map

**Usage**

```r
add_circle_layer(map, source = NULL, filter = NULL,
circle_blur = NULL, circle_color = NULL, circle_opacity = NULL,
circle_pitch_alignment = NULL, circle_pitch_scale = 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 = NULL, visibility = TRUE, popup = NULL,
id = "circle-layer")
```
**add_circle_layer**

Arguments

- **map**
  A mapboxer object.

- **source**
  A Mapbox source. Uses the source from the mapboxer object if no source is supplied.

- **filter**
  A filter expression that is applied to the source.

- **circle_blur**
  (paint) Amount to blur the circle. 1 blurs the circle such that only the centerpoint is full opacity.

- **circle_color**
  (paint) The fill color of the circle.

- **circle_opacity**
  (paint) The opacity at which the circle will be drawn.

- **circle_pitch_alignment**
  (paint) Orientation of circle when map is pitched. One of "map", "viewport".

- **circle_pitch_scale**
  (paint) Controls the scaling behavior of the circle when the map is pitched. One of "map", "viewport".

- **circle_radius**
  (paint) The radius of the circle.

- **circle_sort_key**
  (layout) Sorts features in ascending order based on this value. Features with a higher sort key will appear above features with a lower sort key.

- **circle_stroke_color**
  (paint) The stroke color of the circle.

- **circle_stroke_opacity**
  (paint) The opacity of the circle’s stroke.

- **circle_stroke_width**
  (paint) The width of the circle’s stroke. Strokes are placed outside of the circle_radius.

- **circle_translate**
  (paint) The geometry’s offset. Values are [x, y] where negatives indicate left and up, respectively.

- **circle_translate_anchor**
  (paint) Controls the frame of reference for circle_translate. One of "map", "viewport".

- **visibility**
  (layout) Whether the layer should be displayed.

- **popup**
  A mustache template in which the tags refer to the properties of the layer’s data object.

- **id**
  The unique id of the layer.

See Also

[docs.mapbox.com/mapbox-gl-js/style-spec/layers/#circle](https://docs.mapbox.com/mapbox-gl-js/style-spec/layers/#circle)

Examples

```r
map <- as_mapbox_source(motor_vehicle_collisions_nyc) %>%
  mapboxer() %>%
  set_view_state(-73.9165, 40.7114, 11) %>%
```
add_control

Add a standard control to the map

### Description

Add a standard control to the map

### Usage

```r
add_control(map, control_name, ..., pos = NULL)
add_navigation_control(map, ..., pos = NULL)
add_scale_control(map, ..., pos = NULL)
add_fullscreen_control(map, pos = NULL)
```

### Arguments

- `map` A `mapboxer` object.
- `control_name` The (class) name of the control.
- `...` The options of the control.
- `pos` The position of the control. One of `top-left`, `top-right`, `bottom-right` or `bottom-left`.

### See Also

[https://docs.mapbox.com/mapbox-gl-js/api/markers/](https://docs.mapbox.com/mapbox-gl-js/api/markers/) for available options for the used control.

### Examples

```r
map <- mapboxer() %>%
  add_navigation_control(
    pos = "top-left",
    showCompass = FALSE
  ) %>%
  addfullscreen_control() %>%
  add_scale_control(
    unit = "nautical"
  )
```
### add_draw_control

Add a draw control to the map (experimental)

**Description**

Add a draw control to the map (experimental)

**Usage**

```r
add_draw_control(map, ..., pos = NULL, data = NULL)
```

**Arguments**

- `map`  
  A `mapboxer` object.
- `...`  
  The options of the control.
- `pos`  
  The position of the control. One of `top-left`, `top-right`, `bottom-right` or `bottom-left`.
- `data`  
  A GeoJSON or sf object.

**See Also**


### add_fill_layer

Add a fill layer to the map

**Description**

Add a fill layer to the map

**Usage**

```r
add_fill_layer(map, source = NULL, filter = NULL, 
fill_antialias = TRUE, fill_color = NULL, fill_opacity = NULL, 
fill_outline_color = NULL, fill_pattern = NULL, 
fill_sort_key = NULL, fill_translate = NULL, 
fill_translate_anchor = NULL, visibility = TRUE, popup = NULL, 
id = "fill-layer")
```
**Arguments**

- **map**: A mapboxer object.
- **source**: A Mapbox source. Uses the source from the mapboxer object if no source is supplied.
- **filter**: A filter expression that is applied to the source.
- **fill_antialias**: (paint) Whether or not the fill should be antialiased.
- **fill_color**: (paint) The color of the filled part of this layer. This color can be specified as rgba with an alpha component and the color's opacity will not affect the opacity of the 1px stroke, if it is used.
- **fill_opacity**: (paint) The opacity of the entire fill layer. In contrast to the fill_color, this value will also affect the 1px stroke around the fill, if the stroke is used.
- **fill_outline_color**: (paint) The outline color of the fill. Matches the value of fill_color if unspecified.
- **fill_pattern**: (paint) Name of image in sprite to use for drawing image fills.
- **fill_sort_key**: (layout) Sorts features in ascending order based on this value. Features with a higher sort key will appear above features with a lower sort key.
- **fill_translate**: (paint) The geometry’s offset. Values are [x, y] where negatives indicate left and up, respectively.
- **fill_translate_anchor**: (paint) Controls the frame of reference for fill_translate. One of "map", "viewport".
- **visibility**: (layout) Whether the layer should be displayed.
- **popup**: A mustache template in which the tags refer to the properties of the layer’s data object.
- **id**: The unique id of the layer.

**See Also**

https://docs.mapbox.com/mapbox-gl-js/style-spec/layers/#fill

**Examples**

```r
map <- as_mapbox_source(geojsonsf::geo_melbourne) %>%
  mapboxer() %>%
  set_view_state(
    lng = 144.9624,
    lat = -37.8105,
    zoom = 10,
    pitch = 35
  ) %>%
  add_fill_layer(
    fill_color = c("get", "fillColor"),
    fill_opacity = 0.6,
    popup = "Area: {{AREASQKM}} km<sup>2</sup>",
    # AREASQKM > 5
```
add_filter_control

```r
    filter = list(">", c("get", "AREASQKM"), 5)
```

if (interactive()) map

---

add_filter_control  Add a filter control to the map

**Description**
Add a filter control to the map

**Usage**
```
add_filter_control(map, layer_id, filter = NULL, pos = NULL, rows = 1, cols = 20)
```

**Arguments**
- **map** A mapboxer object.
- **layer_id** The ID of the layer to which the filter is attached.
- **filter** The initial filter expression.
- **pos** The position of the control. One of `top-left`, `top-right`, `bottom-right` or `bottom-left`.
- **rows** The number of rows of the textarea input.
- **cols** The number of columns of the textarea input.

---

add_layer  Add a layer to the map

**Description**
Adds any kind of layer to the map. The type of the layer is specified by the type property of the layer definition.

**Usage**
```
add_layer(map, style, popup = NULL)
```

**Arguments**
- **map** A mapboxer object.
- **style** A named list that defines the style of the layer. See https://docs.mapbox.com/mapbox-gl-js/style-spec/layers/ for available style options for the used layer type.
- **popup** A mustache template in which the tags refer to the properties of the layer’s data object.
add_line_layer

Add a line layer to the map

Description

Add a line layer to the map

Usage

add_line_layer(map, source = NULL, filter = NULL, line_blur = NULL,
line_cap = NULL, line_color = NULL, line_dasharray = NULL,
line_gap_width = NULL, line_gradient = NULL, line_join = NULL,
line_miter_limit = NULL, line_offset = NULL, line_opacity = NULL,
line_pattern = NULL, line_round_limit = NULL, line_sort_key = NULL,
line_translate = NULL, line_translate_anchor = NULL,
line_width = NULL, visibility = NULL, popup = NULL,
id = "line-layer")
**add_line_layer**

**Arguments**

- **map**
  A `mapboxer` object.

- **source**
  A Mapbox source. Uses the source from the `mapboxer` object if no source is supplied.

- **filter**
  A filter expression that is applied to the `source`.

- **line_blur**
  (paint) Blur applied to the line, in pixels.

- **line_cap**
  (layout) The display of line endings. One of "butt", "round", "square".

- **line_color**
  (paint) The color with which the line will be drawn.

- **line_dasharray**
  (paint) Specifies the lengths of the alternating dashes and gaps that form the dash pattern.

- **line_gap_width**
  (paint) Draws a line casing outside of a line’s actual path. The value indicates the width of the inner gap.

- **line_gradient**
  (paint) Defines a gradient with which to color a line feature. Can only be used with GeoJSON sources that specify `lineMetrics = TRUE`.

- **line_join**
  (layout) The display of lines when joining. One of "bevel", "round", "miter".

- **line_miter_limit**
  (layout) Used to automatically convert miter joins to bevel joins for sharp angles. Requires `line_join` to be "miter".

- **line_offset**
  (paint) The line’s offset. For linear features, a positive value offsets the line to the right, relative to the direction of the line, and a negative value to the left. For polygon features, a positive value results in an inset, and a negative value results in an outset.

- **line_opacity**
  (paint) The opacity at which the line will be drawn.

- **line_pattern**
  (paint) Name of image in sprite to use for drawing image lines.

- **line_round_limit**
  (layout) Used to automatically convert round joins to miter joins for shallow angles.

- **line_sort_key**
  (layout) Sorts features in ascending order based on this value. Features with a higher sort key will appear above features with a lower sort key.

- **line_translate**
  (paint) The geometry’s offset. Values are [x, y] where negatives indicate left and up, respectively.

- **line_translate_anchor**
  (paint) Controls the frame of reference for `line_translate`.

- **line_width**
  (paint) Stroke thickness.

- **visibility**
  (layout) Whether the layer should be displayed.

- **popup**
  A mustache template in which the tags refer to the properties of the layer’s data object.

- **id**
  The unique id of the layer.

**See Also**

[https://docs.mapbox.com/mapbox-gl-js/style-spec/layers/#line](https://docs.mapbox.com/mapbox-gl-js/style-spec/layers/#line)
Examples

```r
map <- as_mapbox_source(geojsonsf::geo_melbourne) %>%
  mapboxer(
    center = c(144.9624, -37.8105),
    zoom = 11,
    pitch = 45
  ) %>%
  add_navigation_control() %>%
  add_line_layer(
    line_color = c("get", "strokeColor"),
    line_width = 2,
    popup = "{{SA2_NAME}}"
  )

if (interactive()) map
```

---

**add_marker**

*Add a single marker to the map*

Description

Add a single marker to the map

Usage

```r
add_marker(map, lng, lat, popup = NULL)
```

Arguments

- **map**: A `mapboxer` object.
- **lng**: The longitude of the marker.
- **lat**: The latitude of the marker.
- **popup**: The popup text (HTML) that is displayed when you click on the marker.

Examples

```r
lng <- -0.09
lat <- 51.5

map <- mapboxer() %>%
  set_view_state(lng, lat) %>%
  add_marker(lng, lat, popup = "You are here!"

if (interactive()) map
```
add_mouse_position_control

Add a mouse position control to the map

Description

Add a mouse position control to the map

Usage

add_mouse_position_control(map, mustache_template = NULL, pos = NULL, css_text = NULL)

Arguments

map A mapboxer object.
mustache_template A mustache template that contains the mustache tags lng and lat.
pos The position of the control. One of top-left, top-right, bottom-right or bottom-left.
css_text A cssText string that will modify the style of the control element.

Examples

map <- mapboxer(zoom = 4) %>%
  add_mouse_position_control(
    mustache_template = "<b>Lng:</b>{{lng}}, <b>Lat:</b>{{lat}}",
    pos = "bottom-left"
  )

if (interactive()) map

add_popups

Add popups to a layer

Description

Usually you will add the popups in the add_layer function by setting the popup parameter.

Usage

add_popups(map, layer_id, popup)
Arguments

map A mapboxer object.
layer_id The ID of the layer to which you want to add the popups.
popup A mustache template in which the tags refer to the properties of the layer’s data object.

Examples

LAYER_ID <- "crashes"
mustache_tpl <- "
<b>Date:</b> {{date}}</br>
<b>Time:</b> {{time}}</br>
<b>Number of persons injured:</b> {{injured}}"

map <- motor_vehicle_collisions_nyc %>%
as_mapbox_source() %>%
mapboxer(
  center = c(-73.9165, 40.7114),
  zoom = 9
) %>%
add_circle_layer(
  circle_color = "red",
  circle_blur = 1,
  filter = list(">", "injured", 0),
  id = LAYER_ID
) %>%
add_popups(
  LAYER_ID,
  popup = mustache_tpl
)

if (interactive()) map

---

add_source Add a Mapbox source to the map

Description

Add a Mapbox source to the map

Usage

add_source(map, source, id = "mapboxer")
**add_text_control**  

Add a text control to the map

**Arguments**

- **map**  
  A mapboxer object.

- **source**  
  A Mapbox source.

- **id**  
  The unique id of the data source.

**Description**

Add a text control to the map

**Usage**

```r
add_text_control(map, text, pos = NULL, css_text = NULL)
```

**Arguments**

- **map**  
  A mapboxer object.

- **text**  
  The text (HTML) that is displayed.

- **pos**  
  The position of the control. One of top-left, top-right, bottom-right or bottom-left.

- **css_text**  
  A cssText string that will modify the style of the control element.

**add_tooltips**  

Add tooltips to a layer

**Description**

Add tooltips to a layer

**Usage**

```r
add_tooltips(map, layer_id, tooltip)
```

**Arguments**

- **map**  
  A mapboxer object.

- **layer_id**  
  The ID of the layer to which you want to add the tooltips.

- **tooltip**  
  A mustache template in which the tags refer to the properties of the layer’s data object.
as_mapbox_source  

Convert a data object to a Mapbox GeoJSON source

Description

Convert a data object to a Mapbox GeoJSON source

Usage

as_mapbox_source(data, ...)

## S3 method for class 'json'
as_mapbox_source(data, ...)

## S3 method for class 'data.frame'
as_mapbox_source(data, lng = "lng", lat = "lat", ...)

## S3 method for class 'sf'
as_mapbox_source(data, ...)

Arguments

data  
A data frame that contains longitudes and latitudes in separate columns or an sf-object.

...  
The properties of the source. See https://docs.mapbox.com/mapbox-gl-js/style-spec/sources for available options for the given source type.

lng  
The name of the column containing the longitudes.

lat  
The name of the column containing the latitudes.

basemaps  

A list of basemap style URLs

Description

A list of basemap style URLs

Usage

basemaps

Format

An object of class list of length 2.
basemap_background_style

Create a background style

Description

Creates a background style that can be used as basemap.

Usage

basemap_background_style(color = "#111", opacity = 1)

Arguments

color The color of the background.
opacity The opacity of the background.

basemap_raster_style

Create a raster style

Description

Creates a raster style that can be used as a basemap.

Usage

basemap_raster_style(tiles = stamen_raster_tiles("terrain"),
                     attribution = NULL)

Arguments

tiles A list of tile URLs.
attrIBUTion The attribution text of the tile layer.
fit_bounds

Fit the map to a bounding box

Description

Fit the map to a bounding box

Usage

fit_bounds(map, bounds, ...)

Arguments

map A mapboxer object.
bounds The bounding box as a vector in [west, south, east, north] order or a bbox object.
... Optional arguments, see https://docs.mapbox.com/mapbox-gl-js/api/map/#map#fitbounds.

mapboxer

Create a mapboxer widget

Description

Create a mapboxer widget

Usage

mapboxer(source = NULL, style = basemaps$Carto$dark_matter, ..., 
width = NULL, height = NULL, element_id = NULL, 
token = Sys.getenv("MAPBOX_API_TOKEN"))

Arguments

source A mapbox_source that is added to the map with the ID MAPBOXER.
style The map's Mapbox style.
... The properties of the map, see https://docs.mapbox.com/mapbox-gl-js/api/map/.
width The width of the widget.
height The height of the widget.
element_id The unique ID of the widget.
token A Mapbox API access token. Only needed if you want to use styles from Mapbox.
mapboxer-shiny

Examples

```r
map <- mapboxer(
  center = c(-73.9165, 40.7114),
  zoom = 10,
  minZoom = 6,
  pitch = 30,
  bearing = 45
)

if (interactive()) map
```

mapboxer-shiny  Shiny bindings for mapboxer

Description

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

Usage

mapboxerOutput(outputId, width = "100\%", height = "400px")

renderMapboxer(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 mapboxer
- **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.

Examples

```r
library(shiny)
library(mapboxer)

LAYER_ID <- "mvc"

view <- basicPage(
  h1("mapboxer"),
  sliderInput(
    "slider",
    "Number of persons injured",
    min = 0,
    max = 100
  )
)
```
mapboxer_proxy

Create a mapboxer proxy object

```r
min = 0,
max = max(motor_vehicle_collisions_nyc$injured),
step = 1,
value = 0
),
checkboxInput("hide", "Hide layer"),
mapboxerOutput("map"),
htmlOutput("datetime")
)

server <- function(input, output) {
  output$map <- renderMapboxer{
    as_mapbox_source(motor_vehicle_collisions_nyc) %>%
    mapboxer(
      center = c(-73.9165, 40.7114),
      zoom = 10,
      style = basemap_raster_style(stamen_raster_tiles())
    ) %>%
    add_circle_layer(
      circle_color = "black",
      popup = "Number of persons injured {{injured}}",
      id = LAYER_ID
    ) %>%
    add_mouse_position_control(
      "Lng: {{lng}}</br>Lat: {{lat}}",
      css_text = "text-align: left; width: 180px;"
    ) %>%
    add_navigation_control(pos = "top-left")
  }

  observeEvent(input$slider, {
    mapboxer_proxy("map") %>%
    set_filter(LAYER_ID, list("==", "injured", input$slider)) %>%
    update_mapboxer()
  })

  observeEvent(input$hide, {
    mapboxer_proxy("map") %>%
    set_layout_property(LAYER_ID, "visibility", !input$hide) %>%
    update_mapboxer()
  })

  output$datetime <- renderText(
    props <- input$map_onclick$props
    sprintf("<p>%s %s</p>", props$date, props$time)
  )
}

if (interactive()) shinyApp(view, server)
```
mapbox_source

Description

Create a mapboxer-like object that can be used to update a mapboxer object that has already been rendered in a Shiny app.

Usage

mapboxer_proxy(shiny_id, session = shiny::getDefaultReactiveDomain())

Arguments

shiny_id
The output ID of the mapboxer object that should be updated.

session
The current Shiny session object. Usually the default value can be used.

See Also

update_mapboxer

mapbox_source

Create a Mapbox source

Description

Create a Mapbox source

Usage

mapbox_source(type, ...)

Arguments

type
The type of the source, e.g. geojson.

... The properties of the source. See https://docs.mapbox.com/mapbox-gl-js/style-spec/sources for available options for the given source type.
motor_vehicle_collisions_nyc

Motor Vehicle Collisions in NYC

Description

Motor Vehicle Collisions in NYC

Usage

motor_vehicle_collisions_nyc

Format

A data frame with 1601 rows and 6 variables, where each row is a Motor Vehicle Collision:

- **date**: occurrence date of collision
- **time**: occurrence time of collision
- **lng**: latitude coordinate for Global Coordinate System, WGS 1984, decimal degrees (EPSG 4326)
- **lat**: longitude coordinate for Global Coordinate System, WGS 1984, decimal degrees (EPSG 4326)
- **injured**: number of persons injured
- **killed**: number of persons killed

Source

https://opendata.cityofnewyork.us/

set_data

Update the data of a Mapbox source

Description

Update the data of a Mapbox source

Usage

set_data(map, data, source_id = NULL, ...)

## S3 method for class 'character'
set_data(map, data, source_id = NULL, ...)

## S3 method for class 'json'
set_data(map, data, source_id = NULL, ...)
## S3 method for class 'data.frame'
set_data(map, data, source_id = NULL, lng = "lng",
lat = "lat", ...)

## S3 method for class 'sf'
set_data(map, data, source_id, ...)

**Arguments**

- **map**: A `mapboxer_proxy` object.
- **data**: A GeoJSON object, an url pointing to an external GeoJSON document, a data frame that contains longitudes and latitudes in separate columns or an sf-object.
- **source_id**: The ID of the source whose data should be updated.
- **...**: unused
- **lng**: The name of the column containing the longitudes.
- **lat**: The name of the column containing the latitudes.

**See Also**

- `df_geojson`
- `sf_geojson`

---

**set_filter**

Set the filter of a layer

**Description**

Set the filter of a layer

**Usage**

`set_filter(map, layer_id, filter)`

**Arguments**

- **map**: A `mapboxer` object.
- **layer_id**: The ID of the layer whose property should be updated.
- **filter**: A filter expression that is applied to the source.
set_layer_properties  Update layer properties

Description
Update layer properties

Usage
set_paint_property(map, layer_id, property, value)
set_layout_property(map, layer_id, property, value)

Arguments
map  A mapboxer object.
layer_id  The ID of the layer whose property should be updated.
property  The name of the property that should be updated.
value  The new value of the property.

Functions
- set_paint_property: Update a paint property of a layer.
- set_layout_property: Update a layout property of a layer.

set_style  Set the style of the map

Description
Set the style of the map

Usage
set_style(map, style)

Arguments
map  A mapboxer object.
style  The map’s Mapbox style.
**set_view_state**

*Set the view state of the map*

**Description**

Set the view state of the map

**Usage**

```r
set_view_state(map, lng, lat, zoom = 9, pitch = 0, bearing = 0)
```

**Arguments**

- `map` A `mapboxer` object.
- `lng` The longitude of the geographical center point of the map.
- `lat` The latitude of the geographical center point of the map.
- `zoom` The zoom level of the map.
- `pitch` The pitch (tilt) of the map.
- `bearing` The bearing (rotation) of the map.

**stamen_raster_tiles**

*Get Stamen raster tile URLs*

**Description**

Get Stamen raster tile URLs

**Usage**

```r
stamen_raster_tiles(theme = c("watercolor"))
```

**Arguments**

- `theme` The theme of the tiles.
update_mapboxer  
Update a mapboxer proxy object in a Shiny app

Description

Update a mapboxer proxy object in a Shiny app

Usage

update_mapboxer(proxy_obj, ...)

Arguments

proxy_obj A mapboxer_proxy object.
...  unused
Index

* datasets
  - basemaps, 14
    - motor_vehicle_collisions_nyc, 20
  - add_circle_layer, 2
  - add_control, 4
  - add_draw_control, 5
  - add_fill_layer, 5
  - add_filter_control, 7
  - add_fullscreen_control(add_control), 4
  - add_layer, 7, 11
  - add_line_layer, 8
  - add_marker, 10
  - add_mouse_position_control, 11
  - add_navigation_control(add_control), 4
  - add_popups, 8, 11
  - add_scale_control(add_control), 4
  - add_source, 12
  - add_text_control, 13
  - add_tooltips, 13
  - as_mapbox_source, 14
  - basemap_background_style, 15
  - basemap_raster_style, 15
  - basemaps, 14
  - df_geojson, 21
  - fit_bounds, 16
  - mapbox_source, 16, 19
  - mapboxer, 3–7, 9–13, 16, 16, 19, 21–23
  - mapboxer-shiny, 17
  - mapboxer_proxy, 18, 21, 24
  - mapboxerOutput(mapboxer-shiny), 17
  - motor_vehicle_collisions_nyc, 20
  - renderMapboxer(mapboxer-shiny), 17
  - set_data, 20
  - set_filter, 21
  - set_layer_properties, 22
  - set_layout_property(set_layer_properties), 22
  - set_paint_property(set_layer_properties), 22
  - set_style, 22
  - set_view_state, 23
  - sf_geojson, 21
  - stamen_raster_tiles, 23
  - update_mapboxer, 19, 24