

# Package: deckglgeoarrow (via r-universe)

July 7, 2026

**Title** Use 'GeoArrow' to Add 'Deck.gl' Layers to a  
'maplibregl'/mapboxgl' Map

**Version** 0.0.2.9000

**Description** Leverages the high-performance 'GeoArrow' memory layout to render potentially very large 'Deck.gl' data layers on a 'maplibregl'/mapboxgl' map created with R package 'mapgl'. The heavy lifting is done on the 'JavaScript' side in the browser using 'deck.gl-geoarrow' (<<https://github.com/geoarrow/deck.gl-geoarrow/>>). Currently provides functions for adding Scatterplot (points), Path (lines) and Polygon (polygons) layers. Has support for data classes from R packages 'wk' and 'sf'. In addition, convenience functions for styling data, tooltips and popups, as well as layer management are provided. Furthermore, remotely hosted 'GeoParquet' and 'GeoArrow' files can be visualised directly in the browser, without the need to first read them into R memory. Only the styling instructions are prepared by the user in R and are then transferred to and applied in the browser as the data arrives.

**License** MIT + file LICENSE

**Depends** R (>= 4.1.0)

**Imports** geoarrow, geoarrowWidget, htmltools, htmlwidgets, nanoarrow,

**Suggests** geos, mapgl, quarto, terra, tinytest, wk

**VignetteBuilder** quarto

**Encoding** UTF-8

**Roxygen** list( packages = ``roxut", markdown = TRUE, roclets =  
c(``collate", ``namespace", ``rd", ``roxut::tests\_roclet"))

**RoxygenNote** 7.3.3

**URL** <https://github.com/r-spatial/deckglgeoarrow>,  
<https://r-spatial.github.io/deckglgeoarrow/>

**BugReports** <https://github.com/r-spatial/deckglgeoarrow/issues>

**SystemRequirements** Quarto command line tool  
(<https://github.com/quarto-dev/quarto-cli>).

**Config/pak/sysreqs** cmake make libuv1-dev libzstd-dev zlib1g-dev

**Repository** <https://r-spatial.r-universe.dev>

**Date/Publication** 2026-07-03 09:46:36 UTC

**RemoteUrl** <https://github.com/r-spatial/deckglgeoarrow>

**RemoteRef** HEAD

**RemoteSha** be3175822d7ee67d9582e40019641058712b2932

## Contents

addGeoArrowPathLayer . . . . .	2
addGeoArrowPolygonLayer . . . . .	4
addGeoArrowScatterplotLayer . . . . .	7
dataAccessors . . . . .	10
extJSLibs . . . . .	12
generateDeckglLayerId . . . . .	12
popupOptions . . . . .	13
renderOptions . . . . .	14
<b>Index</b>	<b>17</b>

---

addGeoArrowPathLayer *Add Deck.gl PathLayer to a `mapgl::maplibre()` or `mapgl::mapboxgl()` map using blazing fast `nanoarrow::write_nanoarrow()` data transfer.*

---

## Description

Add Deck.gl PathLayer to a `mapgl::maplibre()` or `mapgl::mapboxgl()` map using blazing fast `nanoarrow::write_nanoarrow()` data transfer.

## Usage

```
addGeoArrowPathLayer(
  map,
  data,
  file,
  url,
  layer_id = "path",
  geom_column_name = "geometry",
  popup = NULL,
  tooltip = NULL,
  render_options = renderOptions(),
  data_accessors = dataAccessors(),
```

```

    popup_options = popupOptions(),
    tooltip_options = tooltipOptions(),
    ...
)

```

## Arguments

map	the <code>mapgl::maplibre()</code> or <code>mapgl::mapboxgl()</code> map to add the layer to.
data	a sf (MULTI)LINestring object.
file	a valid local file path to a geoarrow or geoparquet file to be added to the map. Ignored if data is supplied.
url	a URL to a remotely hosted geoarrow or geoparquet file to be added to the map. Ignored if data or file is supplied.
layer_id	the layer id.
geom_column_name	the name of the geometry column of the sf object. It is inferred automatically if only one is present.
popup	should a popup be constructed? If TRUE, will create a popup from all available attributes of the feature. Can also be a character vector of column names, on which case the popup will include only those columns. If a single character is supplied, then this will be shown for all features. If NULL (default) or FALSE, no popup will be shown.
tooltip	should a tooltip be constructed? If TRUE, will create a tooltip from all available attributes of the feature. Can also be a character vector of column names, on which case the tooltip will include only those columns. If a single character is supplied, then this will be shown for all features. If NULL (default) or FALSE, no tooltip will be shown.
render_options	a list of <a href="#">renderOptions</a>
data_accessors	a list of <a href="#">dataAccessors</a>
popup_options	a list of <a href="#">popupOptions</a>
tooltip_options	a list of <a href="#">tooltipOptions</a>
...	can be used to pass additional props and parameters to the deck.gl instance. See <a href="#">Details</a> for more info.

## Details

... can be used to pass additional props and parameters to the deck.gl instance for fine-tuning rendering behaviour. For example, we can pass a list called `parameters` with settings that control the GPU pipeline of the deck.gl instance. See <https://luma.gl/docs/api-reference/core/parameters> for a list of available parameters.

By default, all deck.gl layers passed to a `maplibre()` map will be drawn on top of existing ones. It is, however, possible to inject layers into the existing `maplibre` (base) layer stack by using `render_options = renderOptions(beforeId = "<some-existing-layer-id>")` which will plot the current layer underneath "<some-existing-layer-id>". See below for an example.

**Value**

The modified map object with the added path layer.

**Examples**

```
library(wk)
library(mapgl)

style_positron = "https://basemaps.cartocdn.com/gl/positron-gl-style/style.json"

m = maplibre(style = style_positron)

## single wk LINESTRING
ln = wkt("LINESTRING (30 10, 10 30, 40 40)")

m |>
  addGeoArrowPathLayer(
    data = ln
    , data_accessors = dataAccessors(
      getColor = "#ff000080"
      , getWidth = 3
    )
  )

## remote parquet file
## paste url together so CRAN check doesn't complain
base_url = "https://raw.githubusercontent.com/geoarrow/"
data_url = "geoarrow-data/v0.2.0/example/files/example_linestring_native.parquet"
url = paste0(base_url, data_url)

m |>
  addGeoArrowPathLayer(
    url = url
    , geom_column_name = "geometry"
    , data_accessors = dataAccessors(
      getColor = "#0000ff90"
      , getWidth = 5
    )
    , popup = TRUE
    , tooltip = TRUE
  )
```

**addGeoArrowPolygonLayer**


---

*Add Deck.gl PolygonLayer to a `mapgl::maplibre()` or `mapgl::mapboxgl()` map using blazing fast `nanoarrow::write_nanoarrow()` data transfer.*

---

**Description**

Add Deck.gl PolygonLayer to a `mapgl::maplibre()` or `mapgl::mapboxgl()` map using blazing fast `nanoarrow::write_nanoarrow()` data transfer.

**Usage**

```
addGeoArrowPolygonLayer(
  map,
  data,
  file,
  url,
  layer_id = "polygon",
  geom_column_name = "geometry",
  popup = NULL,
  tooltip = NULL,
  render_options = renderOptions(),
  data_accessors = dataAccessors(),
  popup_options = popupOptions(),
  tooltip_options = tooltipOptions(),
  ...
)
```

**Arguments**

<code>map</code>	the <code>mapgl::maplibre()</code> or <code>mapgl::mapboxgl()</code> map to add the layer to.
<code>data</code>	a sf (MULTI)POLYGON object.
<code>file</code>	a valid local file path to a geoarrow or geoparquet file to be added to the map. Ignored if data is supplied.
<code>url</code>	a URL to a remotely hosted geoarrow or geoparquet file to be added to the map. Ignored if data or file is supplied.
<code>layer_id</code>	the layer id.
<code>geom_column_name</code>	the name of the geometry column of the sf object. It is inferred automatically if only one is present.
<code>popup</code>	should a popup be constructed? If TRUE, will create a popup from all available attributes of the feature. Can also be a character vector of column names, on which case the popup will include only those columns. If a single character is supplied, then this will be shown for all features. If NULL (default) or FALSE, no popup will be shown.
<code>tooltip</code>	should a tooltip be constructed? If TRUE, will create a tooltip from all available attributes of the feature. Can also be a character vector of column names, on which case the tooltip will include only those columns. If a single character is supplied, then this will be shown for all features. If NULL (default) or FALSE, no tooltip will be shown.
<code>render_options</code>	a list of <a href="#">renderOptions</a>
<code>data_accessors</code>	a list of <a href="#">dataAccessors</a>

popup\_options a list of [popupOptions](#)  
 tooltip\_options a list of [tooltipOptions](#)  
 ... can be used to pass additional props and parameters to the deck.gl instance. See [Details](#) for more info.

### Details

... can be used to pass additional props and parameters to the deck.gl instance for fine-tuning rendering behaviour. For example, we can pass a list called `parameters` with settings that control the GPU pipeline of the deck.gl instance. See <https://luma.gl/docs/api-reference/core/parameters> for a list of available parameters.

By default, all deck.gl layers passed to a `maplibre()` map will be drawn on top of existing ones. It is, however, possible to inject layers into the existing maplibre (base) layer stack by using `render_options = renderOptions(beforeId = "<some-existing-layer-id>")` which will plot the current layer underneath "<some-existing-layer-id>". See below for an example.

### Value

The modified map object with the added polygon layer.

### Examples

```
library(wk)
library(mapgl)

style_positron = "https://basemaps.cartocdn.com/gl/positron-gl-style/style.json"

m = maplibre(style = style_positron)

## single wk POLYGON
pl = wkt("POLYGON ((30 10, 10 30, 40 40, 30 10))")

m |>
  addGeoArrowPolygonLayer(
    data = pl
    , render_options = renderOptions(
      beforeId = "water"
    )
  )

## remote parquet file
## paste url together so CRAN check doesn't complain
base_url = "https://raw.githubusercontent.com/geoarrow/"
data_url = "geoarrow-data/v0.2.0/natural-earth/files/natural-earth_countries_native.parquet"
url = paste0(base_url, data_url)

m |>
  addGeoArrowPolygonLayer(
    url = url
    , geom_column_name = "geometry"
```

```

    , render_options = renderOptions(
      extruded = FALSE
    , stroked = TRUE
    )
  , popup = TRUE
  , tooltip = TRUE
)

```

---

```
addGeoArrowScatterplotLayer
```

*Add Deck.gl ScatterplotLayer to a `mapgl::maplibre()` or `mapgl::mapboxgl()` map using blazing fast `nanoarrow::write_nanoarrow()` data transfer.*

---

## Description

Add Deck.gl ScatterplotLayer to a `mapgl::maplibre()` or `mapgl::mapboxgl()` map using blazing fast `nanoarrow::write_nanoarrow()` data transfer.

## Usage

```

addGeoArrowScatterplotLayer(
  map,
  data,
  file,
  url,
  layer_id = "scatter",
  geom_column_name = "geometry",
  popup = NULL,
  tooltip = NULL,
  render_options = renderOptions(),
  data_accessors = dataAccessors(),
  popup_options = popupOptions(),
  tooltip_options = tooltipOptions(),
  ...
)

```

## Arguments

<code>map</code>	the <code>mapgl::maplibre()</code> or <code>mapgl::mapboxgl()</code> map to add the layer to.
<code>data</code>	a sf (MULTI)POINT object.
<code>file</code>	a valid local file path to a geoarrow or geoparquet file to be added to the map. Ignored if data is supplied.
<code>url</code>	a URL to a remotely hosted geoarrow or geoparquet file to be added to the map. Ignored if data or file is supplied.
<code>layer_id</code>	the layer id.

geom_column_name	the name of the geometry column of the sf object. It is inferred automatically if only one is present.
popup	should a popup be constructed? If TRUE, will create a popup from all available attributes of the feature. Can also be a character vector of column names, on which case the popup will include only those columns. If a single character is supplied, then this will be shown for all features. If NULL (default) or FALSE, no popup will be shown.
tooltip	should a tooltip be constructed? If TRUE, will create a tooltip from all available attributes of the feature. Can also be a character vector of column names, on which case the tooltip will include only those columns. If a single character is supplied, then this will be shown for all features. If NULL (default) or FALSE, no tooltip will be shown.
render_options	a list of <a href="#">renderOptions</a>
data_accessors	a list of <a href="#">dataAccessors</a>
popup_options	a list of <a href="#">popupOptions</a>
tooltip_options	a list of <a href="#">tooltipOptions</a>
...	can be used to pass additional props and parameters to the deck.gl instance. See <a href="#">Details</a> for more info.

### Details

... can be used to pass additional props and parameters to the deck.gl instance for fine-tuning rendering behaviour. For example, we can pass a list called `parameters` with settings that control the GPU pipeline of the deck.gl instance. See <https://luma.gl/docs/api-reference/core/parameters> for a list of available parameters.

By default, all deck.gl layers passed to a `maplibre()` map will be drawn on top of existing ones. It is, however, possible to inject layers into the existing maplibre (base) layer stack by using `render_options = renderOptions(beforeId = "<some-existing-layer-id>")` which will plot the current layer underneath "<some-existing-layer-id>". See below for an example.

### Value

The modified map object with the added scatterplot layer.

### Examples

```
library(wk)
library(mapgl)

style_positron = "https://basemaps.cartocdn.com/gl/positron-gl-style/style.json"

m = maplibre(style = style_positron)

## single wk POINT
pt = wkt("POINT (0 0)")
```

```

m |>
  addGeoArrowScatterplotLayer(
    data = pt
    , layer_id = "wk_point"
  )

## wk POINT data frame
n = 5e3

pts = xy(
  x = runif(n, -180, 180)
  , y = runif(n, -50, 50)
  , crs = 4326
)

dat = data.frame(
  id = 1:length(pts)
  , geometry = pts
)

dat$fillColor = sample(hcl.colors(n, alpha = sample(seq(0, 1, length.out = n))))
dat$lineColor = sample(
  hcl.colors(n, alpha = sample(seq(0, 1, length.out = n)), palette = "inferno")
)
dat$radius = sample.int(15, nrow(dat), replace = TRUE)
dat$lineWidth = sample.int(5, nrow(dat), replace = TRUE)

m = maplibre(
  style = style_positron
) |>
  add_navigation_control(visualize_pitch = TRUE) |>
  add_globe_control()

m |>
  addGeoArrowScatterplotLayer(
    data = dat
    , layer_id = "wk-points-layer"
    , geom_column_name = "geometry"
    , render_options = renderOptions()
    , data_accessors = dataAccessors(
      getRadius = "radius"
      , getFillColor = "fillColor"
      , getLineWidth = "lineWidth"
      , getLineColor = "lineColor"
    )
    , popup = TRUE
    , popup_options = popupOptions(anchor = "bottom-right")
    , tooltip = TRUE
    , tooltip_options = tooltipOptions(anchor = "top-left")
  )

## same as above, but using `beforeId` to inject layer into base layer stack
m |>

```

```

addGeoArrowScatterplotLayer(
  data = dat
  , layer_id = "wk-points-layer-before-water"
  , geom_column_name = "geometry"
  , render_options = renderOptions(beforeId = "water")
  , data_accessors = dataAccessors(
    getRadius = "radius"
    , getFillColor = "fillColor"
    , getLineWidth = "lineWidth"
    , getLineColor = "lineColor"
  )
  , popup = TRUE
  , popup_options = popupOptions(anchor = "bottom-right")
  , tooltip = FALSE
  , tooltip_options = tooltipOptions(anchor = "top-left")
)

## remote parquet file
## paste url together so CRAN check doesn't complain
base_url = "https://raw.githubusercontent.com/geoarrow/"
data_url = "geoarrow-data/v0.2.0/natural-earth/files/natural-earth_cities_native.parquet"
url = paste0(base_url, data_url)

m |>
  addGeoArrowScatterplotLayer(
    url = url
    , layer_id = "parquet-layer"
    , geom_column_name = "geometry"
    , data_accessors = dataAccessors(
      getRadius = 10
      , getFillColor = '#ff000090'
      , getLineColor = '#000000ff'
    )
    , tooltip = TRUE
  )

```

---

dataAccessors

*Deck.gl data accessors*

---

## Description

In deck.gl every layer type has a specific set of data accessors, see e.g. those for [ScatterPlotLayer](#). This function sets all defaults for all available layer functions in this package.

## Usage

```
dataAccessors(...)
```

## Arguments

... named accessors to be passed to the relevant deck.gl JavaScript Method.

## Details

Please refer to the relevant [deck.gl documentation](#) for a more detailed description of the available layer functions. See Details for a list of currently available accessors, their defaults and the layer types they apply to.

If you want to map a certain accessor to a data specific value, you will need to add it to the data and provide the column name to the respective data accessor.

Currently, the following accessors are automatically set to the following defaults:

- `getRadius = 1` (ScatterplotLayer)
- `getColor = c(0, 0, 0, 255)` (ScatterplotLayer, PathLayer)
- `getFillColor = c(0, 0, 0, 130)` (ScatterplotLayer, PolygonLayer)
- `getLineColor = c(0, 0, 0, 255)` (ScatterplotLayer, PolygonLayer)
- `getLineWidth = 1` (ScatterplotLayer, PolygonLayer)
- `getElevation = 1000` (PolygonLayer)
- `getWidth = 1` (PathLayer)

### NOTE:

- accessors `getPosition`, `getPath`, `getPolygon` are handled internally and should not be set!
- all `get*Color` accessors will accept either a vector of `rgb(a)` integers (0-255) or a hex color string (potentially also with alpha) - see examples.

## Value

List with named accessors, possibly modified via ... argument.

## Examples

```
# default accessors
dataAccessors()

# modify selected accessors
dataAccessors(
  getFillColor = c(0, 0, 255, 130),
  getLineColor = "#ff00ffaa"
)
```

---

 extJSLibs

*Names and versions of external JavaScript libraries.*


---

**Description**

Names and versions of the external JavaScript libraries used in deckglgeoarrow.

**Usage**

```
extJSLibs()
```

**Details**

See e.g. <https://cdn.jsdelivr.net/npm/deck.gl/package.json> or <https://cdn.jsdelivr.net/npm/@geoarrow/deck.gl-layers/package.json> for more details on the JavaScript dependencies.

**Value**

A named character vector with the versions of the Deck.gl and geoarrow-deckgl-layers JavaScript libraries shipped with this package.

**Examples**

```
extJSLibs()
```

---

 generateDeckglLayerId *Generate proper internal layer IDs*


---

**Description**

Deck.gl injects layers into maplibre's canvas if `interleaved = TRUE` (the default in all layer functions provided here). To do so, it generates specific layer IDs from the `layer_id` provided. This function generates these deck.gl specific layer IDs on the R side, so they can be used in other controls, such as [add\\_layers\\_control](#).

**Usage**

```
generateDeckglLayerId(layer_id, beforeId = NULL)
```

**Arguments**

<code>layer_id</code>	the layer id provided to the respective <code>addGeoArrowDeckgl*</code> layer function used.
<code>beforeId</code>	the <code>beforeId</code> used in the respective <code>addGeoArrowDeckgl*</code> layer function used.

**Value**

Character vector of internally used layer IDs.

**Examples**

```
generateDeckglLayerId("my_scatterplot_layer")
generateDeckglLayerId(beforeId = "water")
```

---

popupOptions

*Options for popups and tooltips*

---

**Description**

Options for popups and tooltips

**Usage**

```
popupOptions(...)
```

```
tooltipOptions(...)
```

**Arguments**

...            named options to be passed to the popups and tooltips of the map. See [maplibregl PopupOptions](#) for details and available options.

**Details**

Both popupOptions and tooltipOptions are passed to the PopupOptions object of the [maplibregl Popup](#) constructor. See [maplibregl PopupOptions](#) for details.

The popupOptions and tooltipOptions in this package only differ in their respective defaults. These are:

For popupOptions

- anchor = "bottom"
- className = "deckglgeoarrow-popup"
- closeButton = TRUE
- closeOnClick = TRUE
- closeOnMove = FALSE
- focusAfterOpen = TRUE
- maxWidth = "none"
- offset = 0
- subpixelPositioning = FALSE

For tooltipOptions

- anchor = "top-left"
- className = "deckglgeoarrow-tooltip"
- closeButton = FALSE
- closeOnClick = FALSE
- closeOnMove = FALSE
- focusAfterOpen = TRUE
- maxWidth = "none"
- offset = 0
- subpixelPositioning = FALSE

### Value

List with named popup/tooltip options, possibly modified via ... argument.

### Functions

- popupOptions(): options for popups
- tooltipOptions(): options for tooltips

### Examples

```
# default
popupOptions()
tooltipOptions()

# modify selected options
tooltipOptions(anchor = "bottom-right", className = "my-css-class-name")
```

---

renderOptions

*Deck.gl render options*

---

### Description

In deck.gl every layer type has a specific set of render options, see e.g. those for [ScatterPlotLayer](#). This function sets all defaults for all available layer functions in this package. Please refer to the relevant [deck.gl documentation](#) for a more detailed description of the available layer functions. See [Details](#) for a list of currently available options, their defaults and the layer types they apply to.

### Usage

```
renderOptions(...)
```

### Arguments

... named options to be passed to the relevant deck.gl JavaScript Method.

### Details

Currently, the following options are automatically set to the following defaults:

#### ScatterplotLayer

- radiusUnits = "pixels"
- radiusScale = 1
- lineWidthUnits = "pixels"
- lineWidthScale = 1
- stroked = TRUE
- filled = TRUE
- radiusMinPixels = 3
- radiusMaxPixels = 15
- lineWidthMinPixels = 0
- lineWidthMaxPixels = 15
- billboard = FALSE
- antialiasing = FALSE

#### PathLayer

- widthUnits = "pixels"
- widthScale = 1
- widthMinPixels = 1
- widthMaxPixels = 5
- capRounded = TRUE
- jointRounded = FALSE
- miterLimit = 4
- billboard = FALSE

#### PolygonLayer

- lineMiterLimit = 4
- extruded = FALSE
- wireframe = TRUE
- elevationScale = 1
- lineJointRounded = FALSE
- lineWidthUnits = "pixels"
- lineWidthScale = 1
- stroked = TRUE

- filled = TRUE
- lineWidthMinPixels = 0
- lineWidthMaxPixels = 15

**All layers**

- beforeId = NULL
- zIndex = 1

zIndex can be used to set layers order if multiple layers are added to a map. Higher values will be plotted on top of lower values. It is ignored, if beforeId is supplied.

**Value**

List with named options, possibly modified via ... argument.

**Examples**

```
# default settings
renderOptions()

# modify selected options
renderOptions(radiusUnits = "meters", radiusScale = 10)
```

# Index

`add_layers_control`, [12](#)  
`addGeoArrowPathLayer`, [2](#)  
`addGeoArrowPolygonLayer`, [4](#)  
`addGeoArrowScatterplotLayer`, [7](#)  
  
`dataAccessors`, [3](#), [5](#), [8](#), [10](#)  
  
`extJSLibs`, [12](#)  
  
`generateDeckglLayerId`, [12](#)  
  
`mapgl::mapboxgl()`, [2–5](#), [7](#)  
`mapgl::maplibre()`, [2–5](#), [7](#)  
  
`nanoarrow::write_nanoarrow()`, [2](#), [4](#), [5](#), [7](#)  
  
`popupOptions`, [3](#), [6](#), [8](#), [13](#)  
  
`renderOptions`, [3](#), [5](#), [8](#), [14](#)  
  
`tooltipOptions`, [3](#), [6](#), [8](#)  
`tooltipOptions (popupOptions)`, [13](#)