Package 'leafgl'

May 8, 2026

Title High-Performance 'WebGl' Rendering for Package 'leaflet'

Version 0.2.4

Description Provides bindings to the 'Leaflet.glify' JavaScript library which extends the 'leaflet' JavaScript library to render large data in the browser using 'WebGl'.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports htmltools, leaflet, sf, yyjsonr, grDevices

Suggests colourvalues, sp, shiny, testthat (>= 2.1.0)

URL https://github.com/r-spatial/leafgl, https://r-spatial.github.io/leafgl/

BugReports https://github.com/r-spatial/leafgl/issues

NeedsCompilation no

Author Tim Appelhans [cre, aut, cph] (ORCID: <https://orcid.org/0000-0002-9824-2707>), Colin Fay [ctb] (ORCID: <https://orcid.org/0000-0001-7343-1846>), Robert Plummer [ctb] (Leaflet.glify plugin), Kent Johnson [ctb], Sebastian Gatscha [ctb], Olivier Roy [ctb]

Maintainer Tim Appelhans <tappelhans@tutamail.com>

Repository CRAN

Date/Publication 2026-04-15 09:10:02 UTC

Contents

`addGlPolylines` *Add Data to a leaflet map using Leaflet.glify*

Description

Leaflet.glify is a WebGL renderer plugin for leaflet. See https://github.com/robertleeplummerjr/Leaflet.glify for details and documentation.

Usage

```
addGlPolylines(
  map,
  data,
  color = cbind(0, 0.2, 1),
  opacity = 0.6,
  group = "glpolylines",
  popup = NULL,
  label = NULL,
  weight = 1,
  layerId = NULL,
  src = FALSE,
  pane = "overlayPane",
  popupOptions = NULL,
  labelOptions = NULL,
  contextMenu = NULL,
  ...
)

addGlPoints(
  map,
  data,
  fillColor = "#0033ff",
  fillOpacity = 0.8,
  radius = 10,
  group = "glpoints",
  popup = NULL,
  label = NULL,
  layerId = NULL,
  src = FALSE,
  pane = "overlayPane",
  popupOptions = NULL,
  labelOptions = NULL,
```

```
  contextMenu = NULL,
  ...
)

addGlPolygons(
  map,
  data,
  color = cbind(0, 0.2, 1),
  fillColor = color,
  fillOpacity = 0.8,
  group = "glpolygons",
  popup = NULL,
  label = NULL,
  layerId = NULL,
  src = FALSE,
  pane = "overlayPane",
  stroke = TRUE,
  popupOptions = NULL,
  labelOptions = NULL,
  contextMenu = NULL,
  ...
)
```

Arguments

map	a map widget object created from leaflet()
data	sf/sp point/polygon/line data to add to the map.
color	Object representing the color. Can be of class integer, character with color names, HEX codes or random characters, factor, matrix, data.frame, list, json or formula. See the examples or makeColorMatrix for more information.
opacity	feature opacity. Numeric between 0 and 1. Note: expect funny results if you set this to < 1.
group	the name of the group the newly created layers should belong to (for clearGroup() and addLayersControl() purposes). Human-friendly group names are permitted–they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g., markers and polygons) can share the same group name.
popup	Object representing the popup. Can be of type character with column names, formula, logical, data.frame or matrix, Spatial, list or JSON. If the length does not match the number of rows in the data, the popup vector is repeated to match the dimension.
label	a character vector of the HTML content for the labels
weight	line width/thickness in pixels for addGlPolylines.
layerId	the layer id
src	whether to pass data to the widget via file attachments.
pane	A string which defines the pane of the layer. The default is "overlayPane".

popupOptions	A Vector of popupOptions() to provide popups
labelOptions	A Vector of labelOptions() to provide label options for each label. Default NULL
contextMenu	a JS function, that is passed to contextmenu. See the example in ./inst/examples/contextmenu.R
...	Used to pass additional named arguments to write_json_str or write_geojson_str & to pass additional arguments to the underlying JavaScript functions. Typical use-cases include setting 'digits' to round the point coordinates or to pass a different 'fragmentShaderSource' to control the shape of the points. Use • 'point' (default) to render circles with a thin black outline • 'simpleCircle' for circles without outline • 'square' for squares without outline Additional arguments could be 'sensitivity', 'sensitivityHover' or 'vertexShaderSource'. See a full list at the Leaflet.glify repository.
fillColor	fill color
fillOpacity	fill opacity
radius	point size in pixels.
stroke	whether to draw stroke along the path (e.g., the borders of polygons or circles)

Functions

- addGlPolylines(): Add Lines to a leaflet map using Leaflet.glify
- addGlPoints(): Add Points to a leaflet map using Leaflet.glify
- addGlPolygons(): Add Polygons to a leaflet map using Leaflet.glify

Shiny Inputs

The objects created with leafgl send input values to Shiny as the user interacts with them. These events follow the pattern input$MAPID_glify_EVENTNAME. The following events are available:

- **Click Events:** input$MAPID_glify_click
- **Mouseover Events:** input$MAPID_glify_mouseover
- **Mouseout Events:** input$MAPID_glify_mouseout

Each event returns a list containing:

- lat: Latitude of the object or mouse cursor
- lng: Longitude of the object or mouse cursor
- id: The layerId, if any
- group: The group name of the object
- data: The properties of the feature

Note

MULTILINESTRINGs and MULTIPOLYGONs are currently not supported! Make sure you cast your data to LINESTRING or POLYGON first using:

- `sf::st_cast(data, "LINESTRING")`
- `sf::st_cast(data, "POLYGON")`

Examples

```
library(leaflet)
library(leafgl)
library(sf)

storms = st_as_sf(atlStorms2005)
cols = heat.colors(nrow(storms))

leaflet() %>%
  addProviderTiles(provider = providers$CartoDB.Positron) %>%
  addGlPolylines(data = storms, color = cols, popup = TRUE, opacity = 1)


library(leaflet)
library(leafgl)
library(sf)

n = 1e5
df1 = data.frame(id = 1:n,
                 x = rnorm(n, 10, 1),
                 y = rnorm(n, 49, 0.8))
pts = st_as_sf(df1, coords = c("x", "y"), crs = 4326)
cols = topo.colors(nrow(pts))

leaflet() %>%
  addProviderTiles(provider = providers$CartoDB.DarkMatter) %>%
  addGlPoints(data = pts, fillColor = cols, popup = TRUE)

library(leaflet)
library(leafgl)
library(sf)

gadm = st_as_sf(gadmCHE)
gadm = st_cast(gadm, "POLYGON")
cols = grey.colors(nrow(gadm))

leaflet() %>%
  addProviderTiles(provider = providers$CartoDB.DarkMatter) %>%
  addGlPolygons(data = gadm, color = cols, popup = TRUE)
```

checkDim *checkDim*

Description

Check the length of the color vector. It must match the number of rows of the dataset.

Usage

```
checkDim(x, data)
```

Arguments

x	The color vector
data	The dataset

checkDimPop *checkDim*

Description

Check the length of the popup vector. It must match the number of rows of the dataset.

Usage

```
checkDimPop(x, data)
```

Arguments

x	The popup vector
data	The dataset

`leafglOutput` *Use leafgl in shiny*

Description

Use leafgl in shiny

Usage

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

renderLeafgl(expr, env = parent.frame(), quoted = TRUE)
```

Arguments

`outputId`	output variable to read from
`width, height`	the width and height of the map
`expr`	An expression that generates an HTML widget
`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.

Details

See leaflet::leafletOutput for details. `renderLeafgl` is only exported for consistency. You can just as well use leaflet::renderLeaflet (see example). `leafglOutput` on the other hand is needed as it will attach all necessary dependencies.

Value

A UI for rendering leafgl

A server function for rendering leafgl

Examples

```
if (interactive()) {
library(shiny)
library(leaflet)
library(leafgl)
library(sf)

n = 1e4
df1 = data.frame(id = 1:n,
    x = rnorm(n, 10, 3),
    y = rnorm(n, 49, 1.8))
pts = st_as_sf(df1, coords = c("x", "y"), crs = 4326)
```

```
m = leaflet() %>%
 addProviderTiles(provider = providers$CartoDB.DarkMatter) %>%
 addGlPoints(data = pts, group = "pts") %>%
 setView(lng = 10.5, lat = 49.5, zoom = 6) %>%
 addLayersControl(overlayGroups = "pts")

ui <- fluidPage(
    leafglOutput("mymap")
)

server <- function(input, output, session) {
    output$mymap <- renderLeaflet(m)
}

shinyApp(ui, server)
}
```

`makeColorMatrix` *makeColorMatrix*

Description

Transform object to rgb color matrix

Usage

```
makeColorMatrix(x, data, palette, ...)
```

Arguments

`x`	Object representing the color. Can be of class integer, numeric, Date, POSIX*, character with color names or HEX codes, factor, matrix, data.frame, list, json or formula.
`data`	The dataset
`palette`	Name of a color palette. If `colourvalues` is installed, it is passed to `colour_values_rgb`. To see all available palettes, please use `colour_palettes`. If `colourvalues` is not installed, the palette is passed to `colorNumeric`.
`...`	Passed to `colour_palettes` or `colorNumeric`.

Examples

```
{
## For Integer/Numeric/Factor
makeColorMatrix(23L)
makeColorMatrix(23)
makeColorMatrix(as.factor(23))
```

```
## For POSIXt / Date
makeColorMatrix(as.POSIXlt(Sys.time(), "America/New_York"), NULL)
makeColorMatrix(Sys.time(), NULL)
makeColorMatrix(Sys.Date(), NULL)

## For matrix/data.frame
makeColorMatrix(cbind(130,1,1), NULL)
makeColorMatrix(matrix(1:99, ncol = 3, byrow = TRUE), data.frame(x=c(1:33)))
makeColorMatrix(data.frame(matrix(1:99, ncol = 3, byrow = TRUE)), data.frame(x=c(1:33)))

## For characters
testdf <- data.frame(
  texts = LETTERS[1:10],
  vals = 1:10,
  vals1 = 11:20
)
makeColorMatrix("red", testdf)
makeColorMatrix("val", testdf)

## For formulaes
makeColorMatrix(~vals, testdf)
makeColorMatrix(~vals1, testdf)

## For JSON
makeColorMatrix(leafgl:::yyson_json_str(data.frame(r = 54, g = 186, b = 1)), NULL)

## For Lists
makeColorMatrix(list(1,2), data.frame(x=c(1,2)))
}
```

---

makePopup *makePopup*

Description

Transform object to popup

Usage

```
makePopup(x, data)
```

Arguments

x	Object representing the popup
data	The dataset

remove *Remove Leaflet.Glify elements from a map*

Description

Remove one or more features from a map, identified by ‘layerId‘; or, clear all features of the given group.

Usage

```
removeGlPoints(map, layerId)

removeGlPolylines(map, layerId)

removeGlPolygons(map, layerId)

clearGlLayers(map)

clearGlGroup(map, group)
```

Arguments

map	a map widget object, possibly created from leaflet() but more likely from leafletProxy()
layerId	character vector; the layer id(s) of the item to remove
group	the name of the group whose members should be removed

Value

the new ‘map‘ object

Index