Package ‘leafgl’

October 13, 2022

Type Package
Title High-Performance ‘WebGl’ Rendering for Package ‘leaflet’
Version 0.1.1
Maintainer Tim Appelhans <tim.appelhans@gmail.com>
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
LazyData true
RoxygenNote 7.0.2
Imports geojsonsf, htmltools, jsonify, leaflet, sf, grDevices
Suggests colourvalues, shiny, testthat (>= 2.1.0)
NeedsCompilation no
Author Tim Appelhans [cre, aut, cph], Colin Fay [ctb] (<https://orcid.org/0000-0001-7343-1846>), Robert Plummer [ctb] (Leaflet.glify plugin), Kent Johnson [ctb], Sebastian Gatscha [ctb]
Repository CRAN
Date/Publication 2020-06-28 10:30:09 UTC

R topics documented:

addGlPolylines ............................................. 2
checkDim ................................................. 4
checkDimPop ............................................. 5
clearGLayers ............................................. 5
leafglOutput ............................................. 6
makeColorMatrix ......................................... 7
makePopup ............................................... 8
removeGLPoints .......................................... 9
removeGLPolygons ...................................... 9
removeGLPolylines ...................................... 9
addGlPolylines

Description

Leaflet.glify is a web gl 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,
  weight = 1,
  layerId = NULL,
  src = FALSE,
  ...
)

addGlPoints(
  map,
  data,
  fillColor = "#0033ff",
  fillOpacity = 0.8,
  radius = 10,
  group = "glpoints",
  popup = NULL,
  layerId = NULL,
  src = FALSE,
  ...
)

addGlPolygons(
  map,
  data,
  color = cbind(0, 0.2, 1),
  fillColor = color,
  fillOpacity = 0.8,
  group = "glpolygons",
  popup = NULL,
  layerId = NULL,
  src = FALSE,
Arguments

- **map**: a leaflet map to add points/polygons to.
- **data**: sf/sp point/polygon 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**: a group name for the feature layer.
- **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 dataset, the popup vector is repeated to match the dimension.
- **weight**: line width/thickness in pixels for `addGlPolylines`.
- **layerId**: the layer id.
- **src**: whether to pass data to the widget via file attachments.
- **...**: Passed to `to_json` for the data coordinates.
- **fillColor**: fill color.
- **fillOpacity**: fill opacity.
- **radius**: point size in pixels.

Details

MULTILINESTRINGs are currently not supported! Make sure you cast your data to LINETSTRING first (e.g. using `sf::st_cast(data, "LINESTRING")`).

MULTIPOLYGONs are currently not supported! Make sure you cast your data to POLYGON first (e.g. using `sf::st_cast(data, "POLYGON")`).

Functions

- `addGlPolylines`: add polylines 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

Examples

```r
if (interactive()) {
  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)
} if (interactive()) {
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)
}

if (interactive()) {
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)
}

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

checkDim
**checkDimPop**

**Usage**

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

**Arguments**

- `x` : The color vector
- `data` : The dataset

**Description**

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

**Usage**

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

**Arguments**

- `x` : The popup vector
- `data` : The dataset

**clearGlLayers**

**Description**

Clear all Glify features

**Usage**

```r
clearGlLayers(map)
```

**Arguments**

- `map` : The map widget
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)
makeColorMatrix

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

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
library(leaflet)
makeColorMatrix("red", breweries91)
makeColorMatrix("blue", breweries91)
makeColorMatrix("#36ba01", breweries91)
makeColorMatrix("founded", data.frame(breweries91))

## For formulae
makeColorMatrix(~founded, breweries91)
makeColorMatrix(~founded + zipcode, breweries91)

## For JSON
library(jsonify)
makeColorMatrix(jsonify::to_json(data.frame(r = 54, g = 186, b = 1)), NULL)

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

---

**Description**

Transform object to popup

**Usage**

`makePopup(x, data)`

**Arguments**

- `x`: Object representing the popup
- `data`: The dataset
**removeGlPoints**

**Description**
Remove points from a map, identified by layerId;

**Usage**
```
removeGlPoints(map, layerId)
```

**Arguments**
- **map** The map widget
- **layerId** The layerId to remove

**removeGlPolygons**

**Description**
Remove polygons from a map, identified by layerId;

**Usage**
```
removeGlPolygons(map, layerId)
```

**Arguments**
- **map** The map widget
- **layerId** The layerId to remove

**removeGlPolylines**

**Description**
Remove lines from a map, identified by layerId;

**Usage**
```
removeGlPolylines(map, layerId)
```

**Arguments**
- **map** The map widget
- **layerId** The layerId to remove
Index

addGlPoints (addGlPolylines), 2
addGlPolygons (addGlPolylines), 2
addGlPolylines, 2

checkDim, 4
checkDimPop, 5
clearGLLayers, 5
colorNumeric, 7
colour_palettes, 7
colour_values_rgb, 7

leafglOutput, 6
leaflet::leafletOutput, 6
leaflet::renderLeaflet, 6

makeColorMatrix, 3, 7
makePopup, 8

removeGlPoints, 9
removeGlPolygons, 9
removeGlPolylines, 9
renderLeafgl (leafglOutput), 6

to_json, 3