Package 'leafem'

July 26, 2020

Type Package

Title 'leaflet' Extensions for 'mapview'

Version 0.1.3

Date 2020-07-19

Maintainer Tim Appelhans <tim.appelhans@gmail.com>

Description Provides extensions for packages 'leaflet' & 'mapdeck',
many of which are used by package 'mapview'.
Focus is on functionality readily available in
Geographic Information Systems such as 'Quantum GIS'. Includes functions
to display coordinates of mouse pointer position, query image values via
mouse pointer and zoom-to-layer buttons. Additionally, provides a feature
type agnostic function to add points, lines, polygons to a map.

License MIT + file LICENSE

URL https://github.com/r-spatial/leafem

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

Depends R (>= 3.1.0)

Imports base64enc, htmltools (>= 0.3), htmlwidgets, leaflet (>=
2.0.1), raster, sf, png

Suggests clipr, gdalUtils, leafgl, mapdeck, plainview, stars

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Tim Appelhans [cre, aut],
Christoph Reudenbach [ctb],
Kenton Russell [ctb],
Jochen Darley [ctb],
Daniel Montague [ctb] (Leaflet.EasyButton plugin),
Lorenzo Busetto [ctb],
Luigi Ranghetti [ctb],

Miles McBain [ctb],
Sebastian Gatscha [ctb],
Björn Harrtell [ctb] (FlatGeobuf plugin),
Daniel Dufour [ctb] (georaster-layer-for-leaflet)

Repository CRAN

Date/Publication 2020-07-26 09:20:02 UTC

R topics documented:

addCopyExtent 2
addExtent 3
addFeatures 4
addFgb 5
addGeoRaster 7
addGeotiff 8
addHomeButton 10
addImageQuery 11
addLocalFile 13
addLogo 15
addMouseCoordinates 16
addRasterRGB 18
addStarsImage 19
addStaticLabels 21
addTileFolder 22
colorOptions 23
garnishMap 23
updateLayersControl 24

Index **26**

`addCopyExtent` *Copy current view extent to the clipboard*

Description

Add JavaScript functioality to enable copying of the current view bouding box to the clipboard. The `copy.btn` argument expects a valid keycode `event.code` such as "KeyE" (the default). Use `https://keycode.info/` to find the approprate codes for your keyboard.

Usage

```
addCopyExtent(map, event.code = "KeyE")
```

Arguments

`map`	a mapview or leaflet object.
`event.code`	the JavaScript event.code for ley strokes.

Examples

```
library(leaflet)

leaflet() %>%
addProviderTiles("CartoDB.Positron") %>%
  addCopyExtent(event.code = "KeyE") %>%
  addMouseCoordinates()

# now click on the map (!) and zoom to anwhere in the map, then press 'e' on
# your keyboard. This will copy the current extent/bounding box as a JSON object
# to your clipboard which can then be parsed with:

# jsonlite::fromJSON(<Ctrl+v>)
```

`addExtent` *Add extent/bbox of spatial objects to a leaflet map*

Description

This function adds the bounding box of a spatial object to a leaflet or mapview map.

Usage

```
addExtent(map, data, ...)
```

Arguments

`map`	A `leaflet` or `mapview` map.
`data`	A `sf` object to be added to the `map`.
`...`	additional arguments passed on to `addFeatures`

Examples

```
library(leaflet)

# Usage in leaflet
leaflet() %>%
  addProviderTiles("OpenStreetMap") %>%
  addExtent(gadmCHE)

leaflet(gadmCHE) %>%
  addProviderTiles("OpenStreetMap") %>%
  addExtent()
```

addFeatures *Type agnositc version of* leaflet::add* *functions.*

Description

Add simple features geometries from sf

Usage

```
addFeatures(map, data, pane = "overlayPane", ...)
```

Arguments

map	A leaflet or mapview map.
data	A sf object to be added to the map.
pane	The name of the map pane for the features to be rendered in.
...	Further arguments passed to the respective leaflet::add* functions. See addCircleMarkers, addPolylines and addPolygons.

Value

A leaflet map object.

Examples

```
library(leaflet)

leaflet() %>% addProviderTiles("OpenStreetMap") %>% addCircleMarkers(data = breweries91)
leaflet() %>% addProviderTiles("OpenStreetMap") %>% addFeatures(data = breweries91)

leaflet() %>% addProviderTiles("OpenStreetMap") %>% addPolylines(data = atlStorms2005)
leaflet() %>% addProviderTiles("OpenStreetMap") %>% addFeatures(atlStorms2005)

leaflet() %>% addProviderTiles("OpenStreetMap") %>% addPolygons(data = gadmCHE)
leaflet() %>% addProviderTiles("OpenStreetMap") %>% addFeatures(gadmCHE)
```

addFgb *Add a flatgeobuf file to leaflet map*

Description

flatgeobuf is a performant binary geo-spatial file format suitable for serving large data. For more details see https://github.com/bjornharrtell/flatgeobuf and the respective documentation for the GDAL/OGR driver at https://gdal.org/drivers/vector/flatgeobuf.html.

In contrast to classical ways of serving data from R onto a leaflet map, flatgeobuf can stream the data chunk by chunk so that rendering of the map is more or less instantaneous. The map is responsive while data is still loading so that popup queries, zooming and panning will work even though not all data has been rendered yet. This makes for a rather pleasant user experience as we don't have to wait for all data to be added to the map before interacting with it.

Usage

```
addFgb(
  map,
  file = NULL,
  url = NULL,
  layerId = NULL,
  group = NULL,
  popup = NULL,
  label = NULL,
  radius = 10,
  stroke = TRUE,
  color = "#03F",
  weight = 5,
  opacity = 0.5,
  fill = FALSE,
  fillColor = NULL,
  fillOpacity = 0.2,
  dashArray = NULL,
  options = NULL,
  className = NULL,
  scale = scaleOptions(),
  ...
)
```

Arguments

map	a mapview or leaflet object.
file	file path to the .fgb file to be added to map. If set, url is ignored.
url	url of the data to be added to map. Only respected if file = NULL.
layerId	the layer id.

group	the group name for the file to be added to map.
popup	either a logical of whether to show the feature properties (fields) in popups or the name of the field to show in popups.
label	name of the field to be shown as a tooltip.
radius	the size of the circle markers.
stroke	whether to draw stroke along the path (e.g. the borders of polygons or circles).
color	stroke color.
weight	stroke width in pixels.
opacity	stroke opacity.
fill	whether to fill the path with fillColor. If fillColor is set, this will be set to TRUE, default is FALSE.
fillColor	fill color. If set, fill will be set to TRUE.
fillOpacity	fill opacity.
dashArray	a string that defines the stroke dash pattern.
options	a list of extra options for tile layers, popups, paths (circles, rectangles, polygons, ...), or other map elements.
className	optional class name for the popup (table). Can be used to define css for the popup.
scale	named list with instructions on how to scale radius, width, opacity, fillOpacity if those are to be mapped to an attribute column.
...	currently not used.

Examples

```
if (interactive()) {
  library(leaflet)
  library(leafem)

  # via URL
 url = "https://raw.githubusercontent.com/bjornharrtell/flatgeobuf/3.0.1/test/data/UScounties.fgb"

  leaflet() %>%
    addTiles() %>%
    leafem:::addFgb(
      url = url
      , group = "counties"
      , label = "NAME"
      , popup = TRUE
      , fill = TRUE
      , fillColor = "blue"
      , fillOpacity = 0.6
      , color = "black"
      , weight = 1
    ) %>%
      addLayersControl(overlayGroups = c("counties")) %>%
      addMouseCoordinates() %>%
```

```
      setView(lng = -105.644, lat = 51.618, zoom = 3)
}
```

---

addGeoRaster *Add stars/raster image to a leaflet map using optimised rendering.*

Description

Add stars/raster image to a leaflet map using optimised rendering.

Usage

```
addGeoRaster(
  map,
  x,
  group = NULL,
  layerId = NULL,
  resolution = 96,
  opacity = 0.8,
  options = leaflet::tileOptions(),
  colorOptions = colorOptions(),
  project = TRUE,
  pixelValuesToColorFn = NULL,
  ...
)
```

Arguments

map	the map to add the raster data to.
x	the stars/raster object to be rendered.
group	he name of the group this raster image should belong to.
layerId	the layerId.
resolution	the target resolution for the simple nearest neighbor interpolation. Larger values will result in more detailed rendering, but may impact performance. Default is 96 (pixels).
opacity	opacity of the rendered layer.
options	options to be passed to the layer. See tileOptions for details.
colorOptions	list defining the palette, breaks and na.color to be used.
project	whether to project the RasterLayer to conform with leaflets expected crs. Defaults to TRUE and things are likely to go haywire if set to FALSE.
pixelValuesToColorFn	optional JS function to be passed to the browser. Can be used to fine tune and manipulate the color mapping. See https://github.com/r-spatial/leafem/issues/25 for some examples.
...	currently not used.

Details

This uses the leaflet plugin 'georaster-layer-for-leaflet' to render raster data. See https://github.com/GeoTIFF/georaster-layer-for-leaflet for details. The clue is that rendering uses simple nearest neighbor interpolation on-the-fly to ensure smooth rendering. This enables handling of larger rasters than with the standard addRasterImage.

Value

A leaflet map object.

Examples

```
if (interactive()) {
  library(leaflet)
  library(leafem)
  library(stars)

  tif = system.file("tif/L7_ETMs.tif", package = "stars")
  x1 = read_stars(tif)
  x1 = x1[, , , 3] # band 3

  leaflet() %>%
    addTiles() %>%
    leafem:::addGeoRaster(
      x1
      , opacity = 1
      , colorOptions = colorOptions(
        palette = grey.colors(256)
      )
    )
}
```

---

addGeotiff *Add a GeoTIFF file to a leaflet map using optimised rendering.*

Description

Add a GeoTIFF file to a leaflet map using optimised rendering.

Usage

```
addGeotiff(
  map,
  file = NULL,
  url = NULL,
  group = NULL,
  layerId = NULL,
```

```
  resolution = 96,
  opacity = 0.8,
  options = leaflet::tileOptions(),
  colorOptions = NULL,
  pixelValuesToColorFn = NULL,
  ...
)
```

Arguments

map	the map to add the raster data to.
file	path to the GeoTIFF file to render.
url	url to the GeoTIFF file to render. Ignored if file is provided.
group	he name of the group this raster image should belong to.
layerId	the layerId.
resolution	the target resolution for the simple nearest neighbor interpolation. Larger values will result in more detailed rendering, but may impact performance. Default is 96 (pixels).
opacity	opacity of the rendered layer.
options	options to be passed to the layer. See tileOptions for details.
colorOptions	list defining the palette, breaks and na.color to be used.
pixelValuesToColorFn	optional JS function to be passed to the browser. Can be used to fine tune and manipulate the color mapping. See examples & https://github.com/r-spatial/leafem/issues/25 for some examples.
...	currently not used.

Details

This uses the leaflet plugin 'georaster-layer-for-leaflet' to render GeoTIFF data. See https://github.com/GeoTIFF/georaster-layer-for-leaflet for details. The GeoTIFF file is read directly in the browser using geotiffjs (https://geotiffjs.github.io/), so there's no need to read data into the current R session. GeoTIFF files can be read from the file system or via url. The clue is that rendering uses simple nearest neighbor interpolation on-the-fly to ensure smooth rendering. This enables handling of larger rasters than with the standard addRasterImage.

Value

A leaflet map object.

Examples

```
if (interactive()) {
  library(leaflet)
  library(leafem)
  library(stars)
```

```
tif = system.file("tif/L7_ETMs.tif", package = "stars")
x1 = read_stars(tif)
x1 = x1[, , , 3] # band 3

tmpfl = tempfile(fileext = ".tif")

write_stars(st_warp(x1, crs = 4326), tmpfl)

myCustomJSFunc = htmlwidgets::JS(
  "
    pixelValuesToColorFn = (raster, colorOptions) => {
      const cols = colorOptions.palette;
      var scale = chroma.scale(cols);

      if (colorOptions.breaks !== null) {
        scale = scale.classes(colorOptions.breaks);
      }
      var pixelFunc = values => {
        let clr = scale.domain([raster.mins, raster.maxs]);
        if (isNaN(values)) return colorOptions.naColor;
        return clr(values).hex();
      };
      return pixelFunc;
    };
  "
)

leaflet() %>%
  addTiles() %>%
  addGeotiff(
    file = tmpfl
    , opacity = 0.9
    , colorOptions = colorOptions(
      palette = grey.colors
      , na.color = "transparent"
    )
    , pixelValuesToColorFn = myCustomJSFunc
  )

}
```

---

addHomeButton *Add a home button / zoom-to-layer button to a map.*

Description

This function adds a button to the map that enables zooming to a provided extent / bbox.

Usage

```
addHomeButton(map, ext, group = "layer", position = "bottomright", add = TRUE)

removeHomeButton(map)
```

Arguments

map	a mapview or leaflet object.
ext	the extent / bbox to zoom to.
group	the name of the group/layer to be zoomed to (or any character string)
position	the position of the button (one of 'topleft', 'topright', 'bottomleft', 'bottomright'). Defaults to 'bottomright'.
add	logical. Whether to add the button to the map (mainly for internal use).

Functions

- removeHomeButton: remove a homeButton from a map

Examples

```
library(leaflet)
library(raster)

## pass a group name only
m <- leaflet() %>%
  addProviderTiles("OpenStreetMap") %>%
  addCircleMarkers(data = breweries91, group = "breweries91") %>%
  addHomeButton(group = "breweries91")
m

## pass a raster extent - group can now be an arbitrary label
m <- leaflet() %>%
  addProviderTiles("OpenStreetMap") %>%
  addCircleMarkers(data = breweries91, group = "breweries91") %>%
  addHomeButton(ext = extent(breweries91), group = "Brew")
m

## remove the button
removeHomeButton(m)
```

---

addImageQuery *Add image query functionality to leaflet/mapview map.*

Description

Add image query functionality to leaflet/mapview map.

Usage

```
addImageQuery(
  map,
  x,
  band = 1,
  group = NULL,
  layerId = NULL,
  project = TRUE,
  type = c("mousemove", "click"),
  digits,
  position = "topright",
  prefix = "Layer",
  className = "",
  ...
)
```

Arguments

map	the map with the RasterLayer to be queried.
x	the RasterLayer that is to be queried.
band	for stars layers, the band number to be queried.
group	the group of the RasterLayer to be queried.
layerId	the layerId of the RasterLayer to be queried. Needs to be the same as supplied in addRasterImage or addStarsImage.
project	whether to project the RasterLayer to conform with leaflets expected crs. Defaults to TRUE and things are likely to go haywire if set to FALSE.
type	whether query should occur on 'mousemove' or 'click'. Defaults to 'mousemove'.
digits	the number of digits to be shown in the display field.
position	where to place the display field. Default is 'topright'.
prefix	a character string to be shown as prefix for the layerId.
className	a character string to append to the control legend.
...	currently not used.

Details

This function enables Raster*/stars objects added to leaflet/mapview maps to be queried. Standard query is on 'mousmove', but can be changed to 'click'. Note that for this to work, the layerId needs to be the same as the one that was set in addRasterImage or addStarsImage. Currently only works for numeric values (i.e. numeric/integer and factor values are supported).

Value

A leaflet map object.

Examples

```
if (interactive()) {
  if (requireNamespace("plainview")) {
    library(leaflet)
    library(plainview)

    leaflet() %>%
      addProviderTiles("OpenStreetMap") %>%
      addRasterImage(poppendorf[[1]], project = TRUE, group = "poppendorf",
                     layerId = "poppendorf") %>%
      addImageQuery(poppendorf[[1]], project = TRUE,
                    layerId = "poppendorf") %>%
      addLayersControl(overlayGroups = "poppendorf")
  }
}
```

`addLocalFile` *Add vector data to leaflet map directly from the file system*

Description

Add vector data to leaflet map directly from the file system

Usage

```
addLocalFile(
  map,
  file,
  layerId = NULL,
  group = NULL,
  popup = NULL,
  label = NULL,
  radius = 10,
  stroke = TRUE,
  color = "#03F",
  weight = 5,
  opacity = 0.5,
  fill = TRUE,
  fillColor = color,
  fillOpacity = 0.2,
  dashArray = NULL,
  options = NULL
)
```

Arguments

map	a mapview or leaflet object.
file	file path to the file to be added to map. NOTE: will be reprojected on-the-fly if not in "longlat".
layerId	the layer id.
group	the group name for the file to be added to map.
popup	either a logical of whether to show the feature properties (fields) in popups or the name of the field to show in popups.
label	name of the field to be shown as a tooltip.
radius	the size of the circle markers.
stroke	whether to draw stroke along the path (e.g. the borders of polygons or circles).
color	stroke color.
weight	stroke width in pixels.
opacity	stroke opacity.
fill	whether to fill the path with color (e.g. filling on polygons or circles).
fillColor	fill color.
fillOpacity	fill opacity.
dashArray	a string that defines the stroke dash pattern.
options	a list of extra options for tile layers, popups, paths (circles, rectangles, polygons, ...), or other map elements.

Examples

```
if (interactive()) {
  library(leafem)
  library(leaflet)
  library(sf)

  destfile = tempfile(fileext = ".gpkg")

  st_write(st_as_sf(gadmCHE), dsn = destfile)

  leaflet() %>%
    addTiles() %>%
    addLocalFile(destfile, popup = TRUE)
}
```

addLogo *add a local or remote image (png, jpg, gif, bmp, ...) to a leaflet map*

Description

This function adds an image to a map. Both local and remote (web) image sources are supported. Position on the map is completely controllable.

Usage

```
addLogo(
  map,
  img,
  alpha = 1,
  src = c("remote", "local"),
  url,
  position = c("topleft", "topright", "bottomleft", "bottomright"),
  offset.x = 50,
  offset.y = 13,
  width = 60,
  height = 60
)
```

Arguments

map	a mapview or leaflet object.
img	the image to be added to the map.
alpha	opacity of the added image.
src	character specifying the source location ("local" for images from the disk, "remote" for web image sources).
url	an optional URL to be opened when clicking on the image (e.g. company's homepage).
position	one of "topleft", "topright", "bottomleft", "bottomright".
offset.x	the offset in x direction from the chosen position (in pixels).
offset.y	the offset in y direction from the chosen position (in pixels).
width	width of the rendered image in pixels.
height	height of the rendered image in pixels.

Examples

```
library(leaflet)
## default position is topleft next to zoom control

img <- "https://www.r-project.org/logo/Rlogo.svg"
leaflet() %>% addTiles() %>% addLogo(img, url = "https://www.r-project.org/logo/")
```

```
## with local image
if (requireNamespace("png")) {
  library(png)

  img <- system.file("img", "Rlogo.png", package="png")
  leaflet() %>% addTiles() %>% addLogo(img, src = "local", alpha = 0.3)

  ## dancing banana gif :-)
  m <- leaflet() %>%
    addTiles() %>%
    addCircleMarkers(data = breweries91)

  addLogo(m, "https://jeroenooms.github.io/images/banana.gif",
          position = "bottomleft",
          offset.x = 5,
          offset.y = 40,
          width = 100,
          height = 100)
}
```

`addMouseCoordinates` *Add mouse coordinate information at top of map.*

Description

This function adds a box displaying the current cursor location (latitude, longitude and zoom level) at the top of a rendered mapview or leaflet map. In case of mapview, this is automatically added. NOTE: The information will only render once a mouse movement has happened on the map.

Usage

```
addMouseCoordinates(map, epsg = NULL, proj4string = NULL, native.crs = FALSE)

removeMouseCoordinates(map)

clip2sfc(x, clipboard = TRUE)
```

Arguments

`map`	a mapview or leaflet object.
`epsg`	the epsg string to be shown.
`proj4string`	the proj4string to be shown.
`native.crs`	logical. whether to use the native crs in the coordinates box.

x	a charcter string with valid longitude and latitude values. Order matters! If missing and clipboard = TRUE (the default) contents will be read from the clipboard.
clipboard	whether to read contents from the clipboard. Default is TRUE.

Details

If style is set to "detailed", the following information will be displayed:

- x: x-position of the mouse cursor in projected coordinates
- y: y-position of the mouse cursor in projected coordinates
- epsg: the epsg code of the coordinate reference system of the map
- proj4: the proj4 definition of the coordinate reference system of the map
- lat: latitude position of the mouse cursor
- lon: longitude position of the mouse cursor
- zoom: the current zoom level

By default, only 'lat', 'lon' and 'zoom' are shown. To show the details about epsg, proj4 press and hold 'Ctrl' and move the mouse. 'Ctrl' + click will copy the current contents of the box/strip at the top of the map to the clipboard, though currently only copying of 'lon', 'lat' and 'zoom' are supported, not 'epsg' and 'proj4' as these do not change with pan and zoom.

Functions

- removeMouseCoordinates: remove mouse coordinates information from a map
- clip2sfc: convert mouse coordinates from clipboard to sfc

Examples

```
library(leaflet)

leaflet() %>%
  addProviderTiles("OpenStreetMap") # without mouse position info
m = leaflet() %>%
  addProviderTiles("OpenStreetMap") %>%
  addMouseCoordinates()

m

removeMouseCoordinates(m)
```

addRasterRGB *Add an RGB image as a layer*

Description

Create a Red-Green-Blue image overlay from a RasterStack / RasterBrick or stars object based on three layers. Three layers (sometimes referred to as "bands" because they may represent different bandwidths in the electromagnetic spectrum) are combined such that they represent the red, green and blue channel. This function can be used to make 'true (or false) color images' from Landsat and other multi-band satellite images. Note, this text is plagirized, i.e. copied from plotRGB. AddRasterRGB and addStarsRGB are aliases.

Usage

```
addRasterRGB(
  map,
  x,
  r = 3,
  g = 2,
  b = 1,
  quantiles = c(0.02, 0.98),
  domain = NULL,
  na.color = "#BEBEBE80",
  ...
)

addStarsRGB(
  map,
  x,
  r = 3,
  g = 2,
  b = 1,
  quantiles = c(0.02, 0.98),
  domain = NULL,
  na.color = "#BEBEBE80",
  ...
)
```

Arguments

map	a map widget object created from 'leaflet()"
x	a 'RasterBrick', 'RasterStack' or 'stars" raster object
r	integer. Index of the Red channel/band, between 1 and nlayers(x)
g	integer. Index of the Green channel/band, between 1 and nlayers(x)
b	integer. Index of the Blue channel/band, between 1 and nlayers(x)

quantiles	the upper and lower quantiles used for color stretching. If set to NULL, stretching is performed basing on 'domain' argument.
domain	the upper and lower values used for color stretching. This is used only if 'quantiles' is NULL. If both 'domain' and 'quantiles' are set to NULL, stretching is applied based on min-max values.
na.color	the color to be used for NA pixels
...	additional arguments passed on to addRasterImage

Author(s)

Tim Appelhans, Luigi Ranghetti

Examples

```
require(raster)
require(stars)
require(plainview)
require(leaflet)

leaflet() %>%
  addTiles(group = "OpenStreetMap") %>%
  addRasterRGB(plainview::poppendorf, 4,3,2, group = "True colours") %>%
  addStarsRGB(st_as_stars(plainview::poppendorf), 5,4,3, group = "False colours") %>%
  addLayersControl(
    baseGroups = c("Satellite"),
    overlayGroups = c("True colours", "False colours"),
  )
```

---

addStarsImage *Add stars layer to a leaflet map*

Description

Add stars layer to a leaflet map

Usage

```
addStarsImage(
  map,
  x,
  band = 1,
  colors = "Spectral",
  opacity = 1,
  attribution = NULL,
  layerId = NULL,
```

```
  group = NULL,
  project = FALSE,
  method = c("auto", "bilinear", "ngb"),
  maxBytes = 4 * 1024 * 1024,
  data = getMapData(map)
)
```

Arguments

map	a mapview or leaflet object.
x	a stars layer.
band	the band number to be plotted.
colors	the color palette (see colorNumeric) or function to use to color the raster values (hint: if providing a function, set na.color to "#00000000" to make NA areas transparent)
opacity	the base opacity of the raster, expressed from 0 to 1
attribution	the HTML string to show as the attribution for this layer
layerId	the layer id
group	the name of the group this raster image should belong to (see the same parameter under addTiles)
project	if TRUE, automatically project x to the map projection expected by Leaflet (EPSG:3857); if FALSE, it's the caller's responsibility to ensure that x is already projected, and that extent(x) is expressed in WGS84 latitude/longitude coordinates
method	the method used for computing values of the new, projected raster image. "bilinear" (the default) is appropriate for continuous data, "ngb" - nearest neighbor - is appropriate for categorical data. Ignored if project = FALSE. See projectRaster for details.
maxBytes	the maximum number of bytes to allow for the projected image (before base64 encoding); defaults to 4MB.
data	the data object from which the argument values are derived; by default, it is the `data` object provided to `leaflet()` initially, but can be overridden

Details

This is an adaption of addRasterImage. See that documentation for details.

Examples

```
library(stars)
library(leaflet)

tif = system.file("tif/L7_ETMs.tif", package = "stars")
x = read_stars(tif)
leaflet() %>%
```

```
addProviderTiles("OpenStreetMap") %>%
addStarsImage(x, project = TRUE)
```

---

addStaticLabels *Add static labels to* leaflet *or* mapview *objects*

Description

Being a wrapper around addLabelOnlyMarkers, this function provides a smart-and-easy solution to add custom text labels to an existing leaflet or mapview map object.

Usage

```
addStaticLabels(map, data, label, group = NULL, layerId = NULL, ...)
```

Arguments

map	A leaflet or mapview object.
data	A sf or Spatial* object used for label placement, defaults to the locations of the first dataset in 'map'.
label	The labels to be placed at the positions indicated by 'data' as character, or any vector that can be coerced to this type.
group	the group of the static labels layer.
layerId	the layerId of the static labels layer.
...	Additional arguments passed to labelOptions.

Value

A labelled **leaflet** map

Author(s)

Florian Detsch, Lorenzo Busetto

See Also

addLabelOnlyMarkers.

Examples

```
## Not run:
## leaflet label display options
library(leaflet)

lopt = labelOptions(noHide = TRUE,
                    direction = 'top',
                    textOnly = TRUE)

## Add labels on a Leaflet map

indata <- sf::st_read(system.file("shape/nc.shp", package="sf"))

leaflet(indata) %>%
  addProviderTiles("OpenStreetMap") %>%
  addFeatures(.) %>%
  addStaticLabels(., label = indata$NAME)

Modify styling -

leaflet(indata) %>%
  addProviderTiles("OpenStreetMap") %>%
  addFeatures(.) %>%
  addStaticLabels(., label = indata$NAME,
                    style = list("color" = "red", "font-weight" = "bold"))


## End(Not run)
```

`addTileFolder` *Add raster tiles from a local folder*

Description

Add tiled raster data pyramids from a local folder that was created with gdal2tiles.py (see https://gdal.org/gdal2tiles.html for details).

Usage

```
addTileFolder(
  map,
  folder,
  tms = TRUE,
  layerId = NULL,
  group = NULL,
  attribution = NULL,
  options = leaflet::tileOptions(),
  data = leaflet::getMapData(map)
)
```

Arguments

map	a mapview or leaflet object.
folder	the (top level) folder where the tiles (folders) reside.
tms	whether the tiles are served as TMS tiles.
layerId	the layer id.
group	the group name for the tile layer to be added to map.
attribution	the attribution text of the tile layer (HTML).
options	a list of extra options for tile layers. See tileOptions for details. When the tiles were created with gdal2tiles.py argument tms needs to be set to TRUE.
data	the data object from which the argument values are derived; by default, it is the data object provided to leaflet() initially, but can be overridden.

---

colorOptions *Color options for addGeoRaster and addGeotiff*

Description

Color options for addGeoRaster and addGeotiff

Usage

```
colorOptions(palette = NULL, breaks = NULL, na.color = "#bebebe22")
```

Arguments

palette	the color palette to use. Can be a set of colors or a color generating function such as the result of colorRampPalette.
breaks	the breaks at which color should change.
na.color	color for NA values (will map to NaN in Javascript).

---

garnishMap *Garnish/decorate leaflet or mapview maps.*

Description

This function provides a versatile interface to add components to a leaflet or mapview map. It takes functions such as "addMouseCoordinates" or addLayersControl and their respective arguments and adds them to the map. Arguments must be named. Functions can be plain or character strings.

Usage

```
garnishMap(map, ...)
```

Arguments

map	a mapview or leaflet object.
...	functions and their arguments to add things to a map.

Examples

```
library(leaflet)

m <- leaflet() %>% addProviderTiles("OpenStreetMap")
garnishMap(m, addMouseCoordinates)

## add more than one with named argument
library(leaflet)

m1 <- garnishMap(m, addScaleBar, addMouseCoordinates,
                 position = "bottomleft")
m1
```

---

updateLayersControl *Update the layer controls when adding layers to an existing map.*

Description

When adding additional base layers or overlay layers to an existing map, `updateLayersControl` will either update the existing layers control or add a new one if map has none.

Usage

```
updateLayersControl(
  map,
  addBaseGroups = character(0),
  addOverlayGroups = character(0),
  position = "topleft",
  ...
)
```

Arguments

map	A `leaflet` or `mapview` map.
addBaseGroups	group names of base layers to be added to layers control.
addOverlayGroups	group names of overlay layers to be added to layers control.
position	position of control: "topleft", "topright", "bottomleft", or "bottomright".
...	Further arguments passed to `addLayersControl`.

Value

A leaflet map object.

Examples

```
library(leaflet)

map = leaflet() %>%
        addProviderTiles("OpenStreetMap", group = "OSM") %>%
        addProviderTiles("CartoDB.DarkMatter", group = "dark") %>%
        addCircleMarkers(data = breweries91, group = "brew")

map # no layers control

map %>%
  updateLayersControl(addBaseGroups = c("OSM", "dark"),
                      addOverlayGroups = "brew")
```

Index

addCircleMarkers, *4*
addCopyExtent, 2
addExtent, 3
addFeatures, *3*, 4
addFgb, 5
addGeoRaster, 7
addGeotiff, 8
addHomeButton, 10
addImageQuery, 11
addLabelOnlyMarkers, *21*
addLayersControl, *23, 24*
addLocalFile, 13
addLogo, 15
addMouseCoordinates, 16
addPolygons, *4*
addPolylines, *4*
addRasterImage, *8, 9*, *12*, *19, 20*
addRasterRGB, 18
addStarsImage, *12*, 19
addStarsRGB *(*addRasterRGB*)*, 18
addStaticLabels, 21
addTileFolder, 22

clip2sfc *(*addMouseCoordinates*)*, 16
colorOptions, 23
colorRampPalette, *23*

garnishMap, 23

labelOptions, *21*

plotRGB, *18*

removeHomeButton *(*addHomeButton*)*, 10
removeMouseCoordinates
*(*addMouseCoordinates*)*, 16

sf, *4*

tileOptions, *7*, *9*, *23*

updateLayersControl, 24