# Package 'leafem'

April 30, 2019

**Type** Package

**Title** 'leaflet' Extensions for 'mapview'

**Version** 0.0.1

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

**Description** Provides extensions for package 'leaflet', 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** htmltools (>= 0.3), htmlwidgets, leaflet (>= 2.0.1), raster, sf, sp

**Suggests** gdalUtils, plainview, png

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.0

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2019-04-30 16:20:09 UTC

# R topics documented:

addFeatures . . . . . 2
addHomeButton . . . . . 3
addImageQuery . . . . . 4
addLogo . . . . . 5
addMouseCoordinates . . . . . 6
garnishMap . . . . . 7

**Index** **9**

---

`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)
```

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, layer.name = "layer", position = "bottomright",
  add = TRUE)

removeHomeButton(map)
```

## Arguments

| | |
|---|---|
| `map` | a mapview or leaflet object. |
| `ext` | the `extent` / `bbox` to zoom to. |
| `layer.name` | the name of the 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)

m <- leaflet() %>%
  addProviderTiles("OpenStreetMap") %>%
  addCircleMarkers(data = breweries91) %>%
  addHomeButton(extent(breweries91), "breweries91")
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", ...)
```

## 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 link{addStrasImage}. |
| 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. |
| ... | 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 link{addStrasImage}. 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")
  }
}
```

---

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)
```

## 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. |

## 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

## Examples

```
library(leaflet)

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

m

removeMouseCoordinates(m)
```

---

`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
```

# Index

addCircleMarkers, *2*
addFeatures, 2
addHomeButton, 3
addImageQuery, 4
addLayersControl, *7*
addLogo, 5
addMouseCoordinates, 6
addPolygons, *2*
addPolylines, *2*
addRasterImage, *4*

bbox, *3*

extent, *3*

garnishMap, 7

removeHomeButton *(*addHomeButton*)*, 3
removeMouseCoordinates
        *(*addMouseCoordinates*)*, 6

sf, *2*