

Package ‘leafletlegend’

July 23, 2021

Type Package

Title Add Custom Legends to 'leaflet' Maps

Version 0.4.0

Description Provides extensions to the 'leaflet' package to customize legends with images, text styling, orientation, sizing, and symbology.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.3.0)

Imports leaflet, htmltools, stats, base64enc, htmlwidgets

RoxygenNote 7.1.1

URL <https://leafletlegend.roh.engineering>,
<https://github.com/tomroh/leafletlegend>

BugReports <https://github.com/tomroh/leafletlegend/issues>

NeedsCompilation no

Author Thomas Roh [aut, cre],
Ricardo Rodrigo Basa [ctb]

Maintainer Thomas Roh <thomas@roh.engineering>

Repository CRAN

Date/Publication 2021-07-23 05:00:02 UTC

R topics documented:

addLeafLegends	2
addLegendImage	7
addLegendSize	9
makeSymbol	12

Index	14
--------------	-----------

Description

Functions for more control over the styling of 'leaflet' legends. The 'leaflet' map is passed through and the output is a 'leaflet' control so that the legends are integrated with other functionality of the API. Style the text of the labels, the symbols used, orientation of the legend items, and sizing of all elements.

Usage

```
addLegendNumeric(
  map,
  pal,
  values,
  title = NULL,
  shape = c("rect", "stadium"),
  orientation = c("vertical", "horizontal"),
  width = 20,
  height = 100,
  bins = 7,
  numberFormat = function(x) { prettyNum(x, format = "f", big.mark = ",", digits =
    3, scientific = FALSE) },
  tickLength = 4,
  tickWidth = 1,
  decreasing = FALSE,
  fillOpacity = 1,
  group = NULL,
  className = "info legend leaflet-control",
  ...
)

addLegendQuantile(
  map,
  pal,
  values,
  title = NULL,
  labelStyle = "",
  shape = c("rect", "circle", "triangle", "plus", "cross", "diamond", "star",
    "stadium"),
  orientation = c("vertical", "horizontal"),
  width = 24,
  height = 24,
  numberFormat = function(x) { prettyNum(x, big.mark = ",", scientific = FALSE,
    digits = 1) },
  opacity = 1,
```

```
    fillOpacity = opacity,
    group = NULL,
    className = "info legend leaflet-control",
    ...
)

addLegendBin(
  map,
  pal,
  values,
  title = NULL,
  labelStyle = "",
  shape = c("rect", "circle", "triangle", "plus", "cross", "diamond", "star",
    "stadium"),
  orientation = c("vertical", "horizontal"),
  width = 24,
  height = 24,
  opacity = 1,
  fillOpacity = opacity,
  group = NULL,
  className = "info legend leaflet-control",
  ...
)

addLegendFactor(
  map,
  pal,
  values,
  title = NULL,
  labelStyle = "",
  shape = c("rect", "circle", "triangle", "plus", "cross", "diamond", "star",
    "stadium"),
  orientation = c("vertical", "horizontal"),
  width = 24,
  height = 24,
  opacity = 1,
  fillOpacity = opacity,
  group = NULL,
  className = "info legend leaflet-control",
  ...
)
```

Arguments

map	a map widget object created from 'leaflet'
pal	the color palette function, generated from colorNumeric
values	the values used to generate colors from the palette function
title	the legend title, pass in HTML to style

shape	shape of the color symbols
orientation	stack the legend items vertically or horizontally
width	in pixels
height	in pixels
bins	an approximate number of tick-marks on the color gradient for the colorNumeric palette
numberFormat	formatting functions for numbers that are displayed e.g. format, prettyNum
tickLength	in pixels
tickWidth	in pixels
decreasing	order of numbers in the legend
fillOpacity	fill opacity of the legend items
group	group name of a leaflet layer group
className	extra CSS class to append to the control, space separated
...	arguments to pass to addControl
labelStyle	character string of style argument for HTML text
opacity	opacity of the legend items

Value

an object from [addControl](#)

Examples

```
library(leaflet)

data(quakes)

# Numeric Legend

numPal <- colorNumeric('viridis', quakes$depth)
leaflet() %>%
  addTiles() %>%
  addLegendNumeric(
    pal = numPal,
    values = quakes$depth,
    position = 'topright',
    title = 'addLegendNumeric (Horizontal)',
    orientation = 'horizontal',
    shape = 'rect',
    decreasing = FALSE,
    height = 20,
    width = 100
  ) %>%
  addLegendNumeric(
    pal = numPal,
    values = quakes$depth,
    position = 'topright',
```

```

    title = htmltools::tags$div('addLegendNumeric (Decreasing)',
    style = 'font-size: 24px; text-align: center; margin-bottom: 5px;'),
    orientation = 'vertical',
    shape = 'stadium',
    decreasing = TRUE,
    height = 100,
    width = 20
  ) %>%
  addLegend(pal = numPal, values = quakes$depth, title = 'addLegend')

# Quantile Legend
# defaults to adding quantile numeric break points

quantPal <- colorQuantile('viridis', quakes$mag, n = 5)
leaflet() %>%
  addTiles() %>%
  addCircleMarkers(data = quakes,
    lat = ~lat,
    lng = ~long,
    color = ~quantPal(mag),
    opacity = 1,
    fillOpacity = 1
  ) %>%
  addLegendQuantile(pal = quantPal,
    values = quakes$mag,
    position = 'topright',
    title = 'addLegendQuantile',
    numberFormat = function(x) {prettyNum(x, big.mark = ',',
    scientific = FALSE, digits = 2)},
    shape = 'circle') %>%
  addLegendQuantile(pal = quantPal,
    values = quakes$mag,
    position = 'topright',
    title = htmltools::tags$div('addLegendQuantile',
    htmltools::tags$br(),
    '(Omit Numbers)'),
    numberFormat = NULL,
    shape = 'circle') %>%
  addLegend(pal = quantPal, values = quakes$mag, title = 'addLegend')

# Factor Legend
# Style the title with html tags, several shapes are supported drawn with svg

quakes[['group']] <- sample(c('A', 'B', 'C'), nrow(quakes), replace = TRUE)
factorPal <- colorFactor('Dark2', quakes$group)
leaflet() %>%
  addTiles() %>%
  addCircleMarkers(
    data = quakes,
    lat = ~ lat,
    lng = ~ long,
    color = ~ factorPal(group),
    opacity = 1,

```

```

    fillOpacity = 1
  ) %>%
addLegendFactor(
  pal = factorPal,
  title = htmltools::tags$div('addLegendFactor', style = 'font-size: 24px; color: red;'),
  values = quakes$group,
  position = 'topright',
  shape = 'triangle',
  width = 50,
  height = 50
) %>%
addLegend(pal = factorPal,
           values = quakes$group,
           title = 'addLegend')

# Bin Legend
# Restyle the text of the labels, change the legend item orientation

binPal <- colorBin('Set1', quakes$mag)
leaflet() %>%
  addTiles() %>%
  addCircleMarkers(
    data = quakes,
    lat = ~ lat,
    lng = ~ long,
    color = ~ binPal(mag),
    opacity = 1,
    fillOpacity = 1
  ) %>%
  addLegendBin(
    pal = binPal,
    values = quakes$mag,
    position = 'topright',
    title = 'addLegendBin',
    labelStyle = 'font-size: 18px; font-weight: bold;',
    orientation = 'horizontal'
  ) %>%
  addLegend(pal = binPal,
            values = quakes$mag,
            title = 'addLegend')

# Group Layer Control
# Works with baseGroups and overlayGroups

leaflet() %>%
  addTiles() %>%
  addLegendNumeric(
    pal = numPal,
    values = quakes$depth,
    position = 'topright',
    title = 'addLegendNumeric',
    group = 'Numeric Data'
  ) %>%

```

```

addLegendQuantile(
  pal = quantPal,
  values = quakes$mag,
  position = 'topright',
  title = 'addLegendQuantile',
  group = 'Quantile'
) %>%
addLegendBin(
  pal = binPal,
  values = quakes$mag,
  position = 'bottomleft',
  title = 'addLegendBin',
  group = 'Bin'
) %>%
addLayersControl(
  baseGroups = c('Numeric Data', 'Quantile'), overlayGroups = c('Bin'),
  position = 'bottomright'
)

```

addLegendImage

Add a Legend with Images

Description

Creates a legend with images that are embedded into a 'leaflet' map so that images do not need to be packaged when saving a 'leaflet' map as HTML. Full control over the label and title style. The 'leaflet' map is passed through and the output is a control so that legend is fully integrated with other functionalities.

Usage

```

addLegendImage(
  map,
  images,
  labels,
  title = "",
  labelStyle = "font-size: 24px; vertical-align: middle;",
  orientation = c("vertical", "horizontal"),
  width = 20,
  height = 20,
  group = NULL,
  className = "info legend leaflet-control",
  ...
)

```

Arguments

map a map widget object created from 'leaflet'

images	path to the image file
labels	labels for each image
title	the legend title, pass in HTML to style
labelStyle	character string of style argument for HTML text
orientation	stack the legend items vertically or horizontally
width	in pixels
height	in pixels
group	group name of a leaflet layer group
className	extra CSS class to append to the control, space separated
...	arguments to pass to addControl

Value

an object from [addControl](#)

Examples

```
library(leaflet)
data(quakes)

quakes1 <- quakes[1:10,]

leafIcons <- icons(
  iconUrl = ifelse(quakes1$mag < 4.6,
    "http://leafletjs.com/examples/custom-icons/leaf-green.png",
    "http://leafletjs.com/examples/custom-icons/leaf-red.png"
  ),
  iconWidth = 38, iconHeight = 95,
  iconAnchorX = 22, iconAnchorY = 94,
  shadowUrl = "http://leafletjs.com/examples/custom-icons/leaf-shadow.png",
  shadowWidth = 50, shadowHeight = 64,
  shadowAnchorX = 4, shadowAnchorY = 62
)

leaflet(data = quakes1) %>% addTiles() %>%
  addMarkers(~long, ~lat, icon = leafIcons, group = 'Quake Leaves') %>%
  addLegendImage(images = c("http://leafletjs.com/examples/custom-icons/leaf-green.png",
    "http://leafletjs.com/examples/custom-icons/leaf-red.png"),
    labels = c('Green', 'Red'), width = 38, height = 95,
    title = htmltools::tags$div('Leaf',
    style = 'font-size: 24px; text-align: center;'),
    position = 'topright',
    group = 'Quake Leaves') %>%
  addLayersControl(overlayGroups = c('Quake Leaves'), position = 'bottomright')

# use raster images with size encodings
height <- sizeNumeric(quakes$depth, baseSize = 40)
width <- height * 38 / 95
```

```

symbols <- icons(
  iconUrl = 'http://leafletjs.com/examples/custom-icons/leaf-green.png',
  iconWidth = width,
  iconHeight = height)
probs <- c(.2, .4, .6, .8)
leaflet(quakes) %>%
  addTiles() %>%
  addMarkers(icon = symbols,
             lat = ~lat, lng = ~long) %>%
  addLegendImage(
    images = rep("http://leafletjs.com/examples/custom-icons/leaf-green.png", 4),
    labels = round(quantile(height, probs = probs), 0),
    width = quantile(height, probs = probs) * 38 / 95,
    height = quantile(height, probs = probs),
    title = htmltools::tags$div(
      'Leaf',
      style = 'font-size: 24px; text-align: center; margin-bottom: 5px;'),
    position = 'topright', orientation = 'vertical')

```

addLegendSize

Add a legend that for the sizing of symbols

Description

Add a legend that for the sizing of symbols

Usage

```

addLegendSize(
  map,
  pal,
  values,
  title = NULL,
  labelStyle = "",
  shape = c("rect", "circle", "triangle", "plus", "cross", "diamond", "star",
            "stadium"),
  orientation = c("vertical", "horizontal"),
  color,
  fillColor,
  strokeWidth = 1,
  opacity = 1,
  fillOpacity = opacity,
  breaks = 5,
  baseSize = 10,
  numberFormat = function(x) { prettyNum(x, big.mark = ",", scientific = FALSE,
    digits = 1) },
  group = NULL,
  className = "info legend leaflet-control",

```

```

    ...
  )

sizeNumeric(values, baseSize)

sizeBreaks(values, breaks, baseSize, ...)

makeSizeIcons(
  values,
  shape = c("rect", "circle", "triangle", "plus", "cross", "diamond", "star",
    "stadium"),
  pal,
  color,
  colorValues,
  fillColor,
  opacity,
  fillOpacity = opacity,
  strokeWidth = 1,
  baseSize,
  ...
)

```

Arguments

map	a map widget object created from 'leaflet'
pal	the color palette function, generated from colorNumeric
values	the values used to generate sizes and if colorValues is not specified and pal is given, then the values are used to generate colors from the palette function
title	the legend title, pass in HTML to style
labelStyle	character string of style argument for HTML text
shape	shape of the color symbols
orientation	stack the legend items vertically or horizontally
color	the color of the legend symbols, if omitted pal is used
fillColor	fill color of symbol
strokeWidth	width of symbol outline
opacity	opacity of the legend items
fillOpacity	fill opacity of the legend items
breaks	an integer specifying the number of breaks or a numeric vector of the breaks
baseSize	re-scaling size in pixels of the mean of the values, the average value will be this exact size
numberFormat	formatting functions for numbers that are displayed e.g. format, prettyNum
group	group name of a leaflet layer group
className	extra CSS class to append to the control, space separated

... arguments to pass to
[addControl](#) for addLegendSize
[pretty](#) for sizeBreaks
[makeSymbol](#) for makeSizeIcons

colorValues the values used to generate color from the palette function

Value

an object from [addControl](#)

Examples

```
library(leaflet)
data("quakes")
quakes <- quakes[1:100,]
numPal <- colorNumeric('viridis', quakes$depth)
sizes <- sizeNumeric(quakes$depth, baseSize = 10)
symbols <- Map(
  makeSymbol,
  shape = 'triangle',
  color = numPal(quakes$depth),
  width = sizes,
  height = sizes
)
leaflet() %>%
  addTiles() %>%
  addMarkers(data = quakes,
             icon = icons(iconUrl = symbols),
             lat = ~lat, lng = ~long) %>%
  addLegendSize(
    values = quakes$depth,
    pal = numPal,
    title = 'Depth',
    labelStyle = 'margin: auto;',
    shape = c('triangle'),
    orientation = c('vertical', 'horizontal'),
    opacity = .7,
    breaks = 5)

# a wrapper for making icons is provided
sizeSymbols <-
makeSizeIcons(
  quakes$depth,
  shape = 'cross',
  pal = numPal,
  color = 'black',
  strokeWidth = 1,
  opacity = .8,
  fillOpacity = .5,
  baseSize = 20
)
leaflet() %>%
```

```

addTiles() %>%
addMarkers(data = quakes,
            icon = sizeSymbols,
            lat = ~lat, lng = ~long) %>%
addLegendSize(
  values = quakes$depth,
  pal = numPal,
  title = 'Depth',
  shape = 'cross',
  orientation = 'horizontal',
  strokeWidth = 1,
  opacity = .8,
  fillOpacity = .5,
  color = 'black',
  baseSize = 20,
  breaks = 5)

# Group layers control
leaflet() %>%
addTiles() %>%
  addLegendSize(
    values = quakes$depth,
    pal = numPal,
    title = 'Depth',
    labelStyle = 'margin: auto;',
    shape = c('triangle'),
    orientation = c('vertical', 'horizontal'),
    opacity = .7,
    breaks = 5,
    group = 'Depth') %>%
  addLayersControl(overlayGroups = c('Depth'))

```

makeSymbol

Create an SVG tag for the symbol

Description

Create an SVG tag for the symbol

Usage

```

makeSymbol(
  shape,
  width,
  height = width,
  color,
  fillColor = color,
  opacity = 1,
  fillOpacity = opacity,

```

```
    ...  
  )  
  
  makeSymbolIcons(  
    shape = c("rect", "circle", "triangle", "plus", "cross", "diamond", "star",  
              "stadium"),  
    color,  
    fillColor = color,  
    opacity,  
    fillOpacity = opacity,  
    strokeWidth = 1,  
    width,  
    height = width,  
    ...  
  )
```

Arguments

shape	the desired shape of the symbol
width	in pixels
height	in pixels
color	color of the symbol
fillColor	fill color of symbol
opacity	opacity of color
fillOpacity	opacity of fillColor
...	arguments to be passed to svg shape tag
strokeWidth	width in pixels of symbol outline

Value

HTML svg element

Index

`addControl`, [4](#), [8](#), [11](#)
`addLeafLegends`, [2](#)
`addLegendBin` (`addLeafLegends`), [2](#)
`addLegendFactor` (`addLeafLegends`), [2](#)
`addLegendImage`, [7](#)
`addLegendNumeric` (`addLeafLegends`), [2](#)
`addLegendQuantile` (`addLeafLegends`), [2](#)
`addLegendSize`, [9](#)

`colorNumeric`, [3](#), [10](#)

`makeSizeIcons` (`addLegendSize`), [9](#)
`makeSymbol`, [11](#), [12](#)
`makeSymbolIcons` (`makeSymbol`), [12](#)

`pretty`, [11](#)

`sizeBreaks` (`addLegendSize`), [9](#)
`sizeNumeric` (`addLegendSize`), [9](#)