Package 'leafletR'

September 12, 2014

Type Package

Title Interactive web-maps based on the Leaflet JavaScript library

Version 0.3-0

Date 2014-09-12

Author Christian Graul, with contributions from Francois Guillem

Maintainer Christian Graul <christian.graul@gmail.com>

Description Display your spatial data on interactive web-maps using the open-source JavaScript library Leaflet. The package provides basic web-mapping functionality to combine vector data and online map tiles from different sources.

License GPL (>= 2)

URL https://github.com/chgrl/leafletR

Depends R (>= 3.0.0), brew, jsonlite

Suggests httr, rgdal, sp

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-12 17:19:37

R topics documented:

`leafletR-package` *Interactive web-maps based on the Leaflet JavaScript library*

Description

Display your spatial data on interactive web-maps using the open-source JavaScript library Leaflet. The package provides basic web-mapping functionality to combine vector data and online map tiles from different sources.

Details

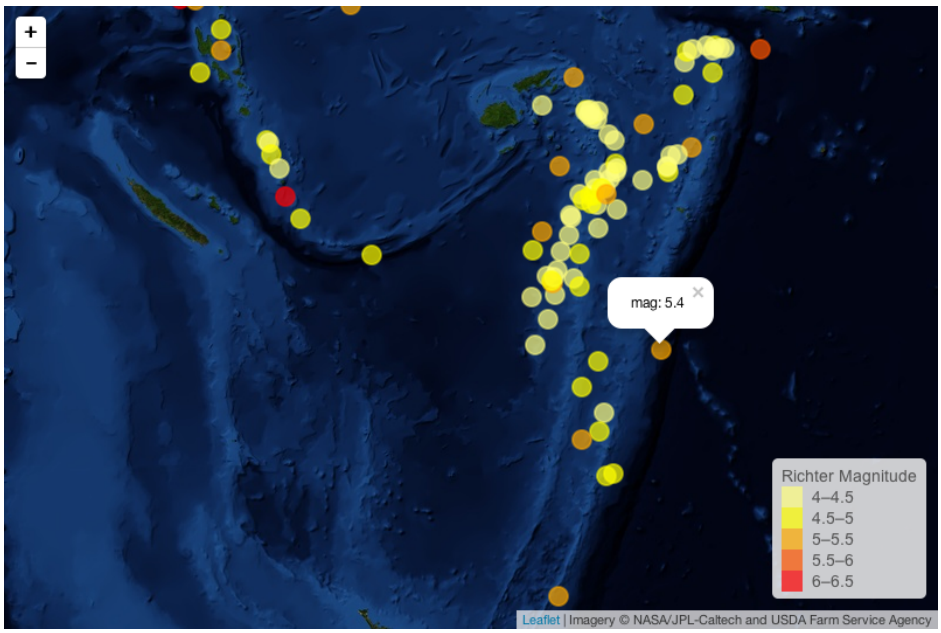
`leafletR` lets you display spatial data on interactive maps in web browsers (a.k.a. slippy maps). It takes advantage of the open-source JavaScript library Leaflet.js, developed by Vladimir Agafonkin. Focusing simplicity, the package provides basic web-mapping functionality and styling options only. For map display an internet connection is required to load the Leaflet library, stylesheets and base map tiles. The ready to use html file output can be viewed locally or uploaded to a web-server.

`leafletR` supports GeoJSON files directly. Additionally it contains conversion tools for sp spatial objects, several popular spatial vector data formats and R data frames containing point coordinates.

`leafletR` uses open base map tiles only. Map data is provided by the OpenStreetMap project and satellite images are provided by courtesy of NASA/ JPL-Caltech and U.S. Department of Agriculture, Farm Service Agency.

Try the example below to check if `leafletR` has been correctly installed. Any question and feedback is welcome via email to <christian.graul@gmail.com> or on GitHub.

Example output:



Author(s)

Christian Graul, with contributions from Francois Guillem

Maintainer: Christian Graul <christian.graul@gmail.com>

References

http://leafletjs.com

Examples

```
# load example data (Fiji Earthquakes)
data(quakes)

# store data in GeoJSON file (just a subset here)
q.dat <- toGeoJSON(data=quakes[1:99,], dest=tempdir(), name="quakes")

# make style based on quake magnitude
q.style <- styleGrad(prop="mag", breaks=seq(4, 6.5, by=0.5),
  style.val=rev(heat.colors(5)), leg="Richter Magnitude",
  fill.alpha=0.7, rad=8)

# create map
q.map <- leaflet(data=q.dat, dest=tempdir(), title="Fiji Earthquakes",
  base.map="mqsat", style=q.style, popup="mag")

# view map in browser
#q.map
```

---

addBaseMap *Add custom base maps*

Description

Add a custom base map to the list of maps available in the function leaflet.

Usage

```
addBaseMap(name, title, url, options)
```

Arguments

name	Name of the base map.
title	Title of the base map, used in the layer control of the resulting map. Optional – if missing, name is used.
url	URL for the base map. See http://leafletjs.com/reference.html#tilelayer for more information.
options	Optional list of additional options. See http://leafletjs.com/reference.html#tilelayer for for a list of valid options.

Author(s)

François Guillem

See Also

leaflet

Examples

```
## Not run:
# duplicates osm base map
addBaseMap(
  name="myosm",
  title="Duplicated OpenStreetMap",
  url="http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
  options=list(
    attribution='&copy; <a href="http://openstreetmap.org/copyright", target=
      "_blank">OpenStreetMap contributors</a>'
  )
)

map <- leaflet(base.map="myosm")

## End(Not run)
```

---

getProperties *Print property names of a GeoJSON file*

Description

Prints the names of all available properties of a GeoJSON file.

Usage

```
getProperties(data, print=TRUE)
prop(data, print=TRUE)
```

Arguments

data	Name of data file as string.
print	If TRUE (default), the property names are printed.

Value

Property names as string vector.

Author(s)

Christian Graul

See Also

styleCat, styleGrad

Examples

```
## Not run:
data(quakes)
qks <- toGeoJSON(data=quakes, dest=tempdir())
getProperties(data=qks)

## End(Not run)
```

`leaflet` *Create a Leaflet web-map*

Description

Creates a web-map of users' spatial data over open-source base maps. Output consists of a ready to use HTML file and a GeoJSON data file.

Usage

```
leaflet(data, dest, title, size, base.map="osm", center,
  zoom, style, popup, incl.data=FALSE, overwrite=TRUE)
leaf(data, dest, title, size, base.map="osm", center,
  zoom, style, popup, incl.data=FALSE, overwrite=TRUE)
```

Arguments

`data`	Name(s) of data file(s) (GeoJSON format), as string or a list of strings. Plotting order follows the file sequence.
`dest`	Path to the `data` file, as string. Optional – if missing, the current working directory is used.
`title`	Map title, as string. Default is `"map"`.
`size`	Size of the map on the website in pixels, as numeric vector – `c(width, height)`. Optional – if missing, a fullscreen (browser window) map is generated.
`base.map`	Base map(s) in the background of the data, as string. One or a list of `"osm"` (OpenStreetMap standard map), `"tls"` (Thunderforest Landscape), `"mqosm"` (MapQuest OSM), `"mqsat"` (MapQuest Open Aerial), `"water"` (Stamen Watercolor) or `"toner"` (Stamen Toner). Default is `"osm"`. If `base.map` is a list, the last item is used as default base map and a layer control button is added to the map.
`center`	Map center coordinates in decimal degrees, as vector of two numeric values: `c(latitude, longitude)`. Optional – if missing, the data layer is centered automatically.

zoom	Map zoom level, as integer value. Usually a value between 0 (global small scale) and 18 (detailed large scale). The MapQuest Open Aerial map (base.map="mqsat") provides only 12 zoom levels [0-11]. Optional – if missing, the zoom level is calculated for the bounding box of the data layer.
style	Style(s) of the data layer(s). One or a list of style object(s), created by styleSingle, styleGrad or styleCat. Optional – if missing, a default style is applied.
popup	Properties (attributes) of the data to be shown in a popup when a map object is clicked. String or a vector of strings. "*" adds all available properties to the popup. A list of (vectors of) strings specifies properties for multiple data layers. Per default no popups are shown.
incl.data	If TRUE, data is included in the HTML file itself. Per default (incl.data=FALSE) the data is saved in a separate file. Including data in the HTML file allows for viewing the map locally on Chrome and Opera browsers.
overwrite	TRUE (which is the default) overwrites existing files with the same name.

Value

HTML file path, as string.

Note

Please note: data only accepts GeoJSON files with one geometry type and geographical coordinates (longlat, WGS84).

Author(s)

Christian Graul

References

Base map tiles are provided by

OpenStreetMap standard map	http://www.openstreetmap.org
Thunderforest Landscape	http://www.thunderforest.com
MapQuest OSM	http://www.mapquest.com
MapQuest Open Aerial	http://www.mapquest.com
Stamen Watercolor	http://stamen.com
Stamen Toner	http://stamen.com

See Also

styleSingle, styleGrad, styleCat

Examples

```
## Not run:
# prepare data
data(quakes)
dat <- toGeoJSON(data=quakes, dest=tempdir())
```

```
# create and view simple map
map <- leaflet(dat)
map  # redirects to browseURL(map)

# set output directory and map title
map <- leaflet(data=dat, dest=tempdir(), title="Fiji Earthquakes")

# set map size, center and zoom level
map <- leaflet(data=dat, dest=tempdir(),
  size=c(800,600), center=c(-18.35, 179.75), zoom=6)

# set base map and popup
# magnitude is used as popup (type names(quakes) for available properties)
map <- leaflet(data=dat, dest=tempdir(),
  base.map="mqsat", popup="mag")

# include data in HTML file
map <- leaflet(dat, incl.data=TRUE)

# preserve existing files from overwriting
map <- leaflet(dat, overwrite=FALSE)

# more than one base map
map <- leaflet(data=dat, dest=tempdir(),
  base.map=list("osm", "mqsat", "tls"))

# multiple properties in the popup
map <- leaflet(data=dat, dest=tempdir(),
  popup=c("mag", "depth"))

# all available properties in the popup
map <- leaflet(data=dat, dest=tempdir(),
  popup="*")

# change style
sty <- styleSingle(col="red", fill=NA)
map <- leaflet(data=dat, dest=tempdir(), base.map="mqsat", style=sty)

# more than one data set
park <- system.file(package="leafletR", "files", "park_sk.geojson")
peak <- toGeoJSON(system.file(package="leafletR", "files", "peak_sk.kmz"),
  dest=tempdir())
sty.1 <- styleSingle(col="green", fill="green")
sty.2 <- styleSingle(col="brown", fill="brown", rad=3)
map <- leaflet(data=list(park, peak), dest=tempdir(),
  style=list(sty.1, sty.2), popup=list("*", "Name"))

map <- leaflet(data=list(National.Parks=park, Peaks=peak), dest=tempdir(),
  style=list(sty.1, sty.2), popup=list("*", "Name"))  # names in legend

## End(Not run)
```

styleCat *Categorized styling*

Description

Creates a categorized style based on an attribute

Usage

```
styleCat(prop, val, style.par, style.val, leg, ...)
cats(prop, val, style.par, style.val, leg, ...)
```

Arguments

prop	Property (attribute) of the data to be styled, as string.
val	A vector giving the data values to be used as categories.
style.par	Styling parameter as string. One of "col" (categorized color) or "rad" (categorized radius). Categorized radius can only be applied to points.
style.val	Styling values, a vector of colors or radii applied to the categories given by val. See details for unspecified data values.
leg	Legend title as string. The line break sequence \n may be used for line splitting.
...	Additional styling parameters, see styleSingle for details.

Details

If val does not cover all data values, the unspecified data values are colored gray. By adding an extra color for unspecified data values to style.val, an "other"-category is shown in the legend.

Value

A categorized style object.

Author(s)

Christian Graul

See Also

styleSingle, styleGrad, leaflet

Examples

```
## Not run:
# prepare data
dat <- system.file(package="leafletR", "files", "park_sk.geojson")

# simple categorizing
sty <- styleCat(prop="lynx", val=c("yes", "no"),
  style.val=c("green", "red"), leg="Lynx occurrence")
map <- leaflet(data=dat, dest=tempdir(), title="Lynx",
  style=sty)

# just one category
sty <- styleCat(prop="wisent", val="yes", style.val="red",
  leg="Wisent occurrence")
map <- leaflet(data=dat, dest=tempdir(), title="Wisent",
  style=sty)

# get nice colors using ColorBrewer
require(RColorBrewer)
pal <- brewer.pal(7, "Dark2")
sty <- styleCat(prop="year", val=c("1949", "1967", "1978", "1988",
  "1997", "1998", "2002"), style.val=pal, leg="established:\n")
map <- leaflet(data=dat, dest=tempdir(),
  title="National parks", style=sty)

# add 'other'-category to legend
require(RColorBrewer)
pal <- brewer.pal(7, "Dark2")
sty <- styleCat(prop="year", val=c("1997", "1998", "2002"),
  style.val=pal, leg="established:\n")
map <- leaflet(data=dat, dest=tempdir(),
  title="National parks", style=sty)

# additional styling parameters
sty <- styleCat(prop="brown_bear", val=c("yes", "no"),
  style.val=c("darkgreen", "red"), leg="Brown bear\noccurrence",
  alpha=1, lwd=4, fill=NA)
map <- leaflet(data=dat, dest=tempdir(), title="Brown bear",
  style=sty)

## End(Not run)
```

`styleGrad` *Graduated styling*

Description

Creates a graduated style based on an attribute.

Usage

```
styleGrad(prop, breaks, right, out, style.par, style.val, leg, ...)
grads(prop, breaks, right, out, style.par, style.val, leg, ...)
```

Arguments

prop	Property (attribute) of the data to be styled, as string.
breaks	A vector giving the breakpoints between the desired classes.
right	If TRUE (default) classes are right-closed (left-open) intervals (>= breakpoint). Otherwise classes are left-closed (right-open) intervals (> breakpoint).
out	Handling of data outside the edges of breaks. One of 0 (left and right-closed), 1 (left-closed, right-open), 2 (left-open, right-closed) or 3 (left and right-open). Default is 0.
style.par	Styling parameter as string. One of "col" (graduated color) or "rad" (graduated radius). Graduated radius can only be applied to points.
style.val	Styling values, a vector of colors or radii applied to the classes.
leg	Legend title as string. The line break sequence \n may be used for line splitting.
...	Additional styling parameters, see styleSingle for details.

Value

A graduated style object.

Author(s)

Christian Graul

See Also

styleSingle, styleCat, leaflet

Examples

```
## Not run:
# prepare data
data(quakes)
qks <- toGeoJSON(data=quakes, dest=tempdir())

# prepare style
range(quakes$mag) # gives 4.0 and 6.4
sty <- styleGrad(prop="mag", breaks=seq(4, 6.5, by=0.5),
  style.val=rev(heat.colors(5)), leg="Richter Magnitude")

# create map
map <- leaflet(data=qks, dest=tempdir(),
  title="Fiji Earthquakes", style=sty)

# left-closed intervals
```

```
# note the gray points on the map: magnitude of 4 is outside the breaks
# (which are >4.0, >4.5, >5.0, >5.5, >6.0 and >6.5)
sty <- styleGrad(prop="mag", breaks=seq(4, 6.5, by=0.5), right=FALSE,
  style.val=rev(heat.colors(5)), leg="Richter Magnitude")
map <- leaflet(data=qks, dest=tempdir(),
  title="Fiji Earthquakes", style=sty)

# handle outliers
# include outliers of the upper example by left-opening the break edges
# and adding a sixth color
sty <- styleGrad(prop="mag", breaks=seq(4, 6.5, by=0.5), right=FALSE,
  out=2, style.val=rev(heat.colors(6)), leg="Richter Magnitude")
map <- leaflet(data=qks, dest=tempdir(),
  title="Fiji Earthquakes", style=sty)

# graduated radius
sty <- styleGrad(prop="mag", breaks=seq(4, 6.5, by=0.5), style.par="rad",
  style.val=c(2,5,9,14,20), leg="Richter Magnitude")
map <- leaflet(data=qks, dest=tempdir(),
  title="Fiji Earthquakes", style=sty)

# additional styling parameters
peak <- toGeoJSON(data=system.file(package="leafletR", "files",
  "peak_sk.kmz"), dest=tempdir())
sty <- styleGrad(prop="Name", breaks=seq(750, 2500, by=250), out=3,
  style.val=terrain.colors(9), leg="Elevation",
  col=NA, fill.alpha=1, rad=3)
map <- leaflet(data=peak, dest=tempdir(), title="Peak elevation",
  base.map="mqsat", style=sty, popup="Name")

## End(Not run)
```

`styleSingle` *Single symbol styling*

Description

Creates a single symbol style.

Usage

```
styleSingle(col, lwd, alpha, fill, fill.alpha, rad)
singles(col, lwd, alpha, fill, fill.alpha, rad)
```

Arguments

`col`	Color used for lines, i.e. lines itself, borders of polygons and circle borders (points). Color might be given as name, number [0-8] or hexadecimal code. If `fill` is not specified, `col` is used for border and circle area. If `col` is `NA`, the border is omitted.

lwd	Line width in number of pixels – default is 2.
alpha	Opacity of a line or border, as numeric value between 0 (fully transparent) and 1 (opaque).
fill	Fill color used for polygons and circles (points). Color might be given as name, number [0-8] or hexadecimal code. If fill is NA, the circle area is left blank.
fill.alpha	Opacity of a polygon or circle area, as numeric value between 0 (fully transparent) and 1 (opaque).
rad	Radius of circles (points), in number of pixels – default is 10.

Value

A single symbol style object.

Note

Points are displayed as circles.

Author(s)

Christian Graul

See Also

styleGrad, styleCat, leaflet

Examples

```
## Not run:
## point data ##
# prepare data
data(quakes)
dat <- toGeoJSON(data=quakes, dest=tempdir())

# change circle borders
# note: if fill color is not specified, col is also used as fill color
sty <- styleSingle(col=2, lwd=1, alpha=1)
map <- leaflet(data=dat, dest=tempdir(), style=sty)

# change fill color, alpha and radius
sty <- styleSingle(fill="red", fill.alpha=1, rad=2)
map <- leaflet(data=dat, dest=tempdir(), style=sty)

# no border
sty <- styleSingle(col=NA)
map <- leaflet(data=dat, dest=tempdir(), style=sty)

# blank circle area
sty <- styleSingle(fill=NA)
map <- leaflet(data=dat, dest=tempdir(), style=sty)
```

```
# change all arguments
sty <- styleSingle(col="#d4d4d4", lwd=1, alpha=0.8,
  fill="darkred", fill.alpha=0.4, rad=4)
map <- leaflet(data=dat, dest=tempdir(), style=sty)


## line data ##
# prepare data
dat <- toGeoJSON(data=system.file(package="leafletR", "files",
  "lynx.zip"), name="Lynx telemetry", dest=tempdir())

# style
sty <- styleSingle(col="#bb650b", lwd=3, alpha=0.8)
map <- leaflet(data=dat, dest=tempdir(), style=sty)


## polygon data ##
# prepare data
dat <- system.file(package="leafletR", "files", "park_sk.geojson")

# change borders
# note: if fill color is not specified, col is also used as fill color
sty <- styleSingle(col=3, lwd=2, alpha=1)
map <- leaflet(data=dat, dest=tempdir(), style=sty)

# change fill color and alpha
sty <- styleSingle(fill="darkgreen", fill.alpha=0.8)
map <- leaflet(data=dat, dest=tempdir(), style=sty)

# no border
sty <- styleSingle(col=NA)
map <- leaflet(data=dat, dest=tempdir(), style=sty)

# blank polygon area
sty <- styleSingle(fill=NA)
map <- leaflet(data=dat, dest=tempdir(), style=sty)

# change all arguments
sty <- styleSingle(col="#006400", lwd=5, alpha=0.8,
  fill="darkgreen", fill.alpha=0.4)
map <- leaflet(data=dat, dest=tempdir(), style=sty)

## End(Not run)
```

---

toGeoJSON *Create GeoJSON file from spatial data*

Description

Creates a GeoJSON file from data frame, Spatial object or an external spatial data file.

Usage

```
toGeoJSON(data, name, dest, lat.lon, overwrite=TRUE)
tg(data, name, dest, lat.lon, overwrite=TRUE)
```

Arguments

data	Spatial data: data.frame with at least two columns, representing the point coordinates, Spatial object (sp package) or path to external spatial data file as string. See below for details.
name	Name of the resulting GeoJSON file, as string. Optional – if missing, the name of the data frame or data file is used.
dest	Directory the file shall be saved to, as string. Optional – if missing, the current working directory is used.
lat.lon	For data frame conversion only. Names or indices of the columns in data containing the coordinates, as vector of two: c(latitude, longitude). Optional – if missing, the first two columns are used.
overwrite	TRUE (which is the default) overwrites existing files with the same name.

Details

toGeoJSON can handle three types of spatial data: a simple data.frame containing point coordinates and optional data columns, Spatial objects and external spatial data files.

Spatial *objects*

Spatial objects (sp package) should have geographical coordinates (longlat, WGS84). If other projections are used, toGeoJSON can transform the coordinates on the fly, using the rgdal package.

Conversion of external spatial data files

toGeoJSON uses the Ogre web API (http://ogre.adc4gis.com). See the Ogre website for a list of supported formats. Please note that for Shapefiles, MapInfo and VRT, Ogre only accepts a zip file. The Ogre API does not support large files (>15 MB). Have a look at the rgdal package and its writeOGR function, to convert files on your local machine.

Value

GeoJSON file path, as string.

Author(s)

Christian Graul

Source

The code for the conversion of external data files is taken from the togeojson function of the rgbif package. Package import would have unreasonably increased the dependencies of leafletR.

See Also

leaflet

Examples

```
## Not run:
# convert data frame
data(quakes)
toGeoJSON(data=quakes, name="quakes", dest=tempdir(), lat.lon=c(1,2))

# convert data frame - minimal call
# storing output file path in variable
data(quakes)
path <- toGeoJSON(data=quakes)

# preserve existing files from overwriting
toGeoJSON(data=quakes, overwrite=FALSE)

# convert Spatial objects
library(sp)
data(meuse)
coordinates(meuse) <- ~x+y
proj4string(meuse) <- CRS("+init=epsg:28992")
toGeoJSON(data=meuse, dest=tempdir()) # rgdal package required

crd <- coordinates(meuse)
msl <- SpatialLines(list(Lines(list(Line(crd)), "line1")),
  proj4string=CRS("+init=epsg:28992"))
toGeoJSON(data=msl, dest=tempdir()) # rgdal package required

data(meuse.riv)
msp <- SpatialPolygons(list(Polygons(list(Polygon(meuse.riv)),
  "meuse.riv")), proj4string=CRS("+init=epsg:28992"))
toGeoJSON(data=msp, dest=tempdir()) # rgdal package required

# convert a shapefile (in zipped archive)
toGeoJSON(data=system.file(package="leafletR", "files", "lynx.zip"),
  name="lynx_telemetry", dest=tempdir())

# convert a KML/KMZ file
# using name of data file and saving to working directory
toGeoJSON(system.file(package="leafletR", "files", "peak_sk.kmz"))

## End(Not run)
```

Index