

# Package ‘rleafmap’

August 29, 2016

**Title** Interactive Maps with R and Leaflet

**Description** Display spatial data with interactive maps powered by the open-source JavaScript library 'Leaflet' (see <<http://leafletjs.com/>>). Maps can be rendered in a web browser or displayed in the HTML viewer pane of 'RStudio'. This package is designed to be easy to use and can create complex maps with vector and raster data, web served map tiles and interface elements.

**Version** 0.2

**Author** Francois Keck <[francois.keck@gmail.com](mailto:francois.keck@gmail.com)>

**Maintainer** Francois Keck <[francois.keck@gmail.com](mailto:francois.keck@gmail.com)>

**Depends** R (>= 3.0.0)

**Imports** knitr (>= 1.5), sp, raster, methods, grDevices, graphics, utils

**License** GPL-3

**URL** <http://www.francoiskeck.fr/rleafmap/>,  
<https://github.com/fkeck/rleafmap>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-09-28 17:02:35

## R topics documented:

basemap . . . . .	2
bmCredit . . . . .	3
bmServer . . . . .	4
bmSource . . . . .	4
campsites . . . . .	5
chunkerize . . . . .	5
hotels . . . . .	6
layerLegend . . . . .	7
print.splpoints . . . . .	8
rleafmap . . . . .	9

spLayer	9
spLayer.default	10
spLayer.SpatialGridDataFrame	10
spLayer.SpatialLines	11
spLayer.SpatialPoints	11
spLayer.SpatialPolygons	12
spLayerControl	13
summary.basemap	14
toGeoJSON	15
toJS	16
ui	16
uiJS	17
velov	17
writeMap	18

## Index 19

---

basemap	<i>Define a Tile Basemap Layer</i>
---------	------------------------------------

---

### Description

Define a new basemap layer from a tile server.

### Usage

```
basemap(URL, name = NULL, alpha = 1, minZoom = 0, maxZoom = 18,
        tileSize = 256, tms = FALSE)
```

### Arguments

URL	a character string giving the tile server url or a the name of a pre-configured server.
name	a character string to name the layer.
alpha	a numeric value in $[0, 1]$ setting the layer opacity.
minZoom, maxZoom	numeric values setting the minimum and maximum zoom level.
tileSize	a numeric value setting tile size (width and height in pixels, assuming tiles are square).
tms	logical. If TRUE, inverses Y axis numbering for tiles (for TMS services)

### Details

URL should have the form 'http://{s}.somedomain.com/somepath/{z}/{x}/{y}.png' with {s} a facultative subdomain, {z} the zoom level and {x}, {y} the coordinates. **rleafmap** comes with a list of pre-configured servers. Names of these servers are returned by the function `bmSource`.

**Value**

An object of class basemap which can be directly used in [writeMap](#).

**See Also**

[spLayer](#) to define data layers.

**Examples**

```
## Not run:  
#A simple map with two nice basemaps.  
bm1 <- basemap("mapquest.map")  
bm2 <- basemap("stamen.watercolor")  
writeMap(bm1, bm2)  
  
## End(Not run)
```

---

bmCredit

*Tiles Servers Attribution*

---

**Description**

Take a tiles server url and return its attribution.

**Usage**

```
bmCredit(x)
```

**Arguments**

x                    a character string of the url of the server.

**Value**

The attribution of the server.

**bmServer***Tiles Servers URL*

---

**Description**

Take a tiles server name (as returned by [bmSource](#)) and return its url.

**Usage**

```
bmServer(x)
```

**Arguments**

x                    a character string of the name of the server.

**Value**

The url of the server.

---

**bmSource***Basemap Tiles Servers*

---

**Description**

Print a list of tiles servers ready-to-use with [basemap](#).

**Usage**

```
bmSource(print.servers = TRUE)
```

**Arguments**

print.servers    logical. Should the names of the servers be printed?

**Value**

Returns invisibly a matrix with servers names, urls and credits.

---

`campsites`*French Campsites*

---

**Description**

This dataset gives the number of ranked campsites and the number of tent pitches for each department of metropolitan France.

**Usage**

```
data(campsites)
```

**Format**

a `SpatialPolygonsDataFrame` with geometries of the 96 french departements (epsg:4326) and 11 variables.

- `DEP.CODE` The code number of each department.
- `DEP.NAME` The name of each department.
- `CHF.NAME` The name of the main (administrative) city of each department.
- `REGION.NAME` The name of the administrative french region of each department.
- `N.CAMPSITES` The number of campsites.
- `N.5, N.4, N.3, N.2, N.1` The number of campsites for each ranking categories (i.e. stars).
- `PITCHES` The number of camp pitches for each department.

**Source**

- Institut National de l'Information Geographique et Forestiere (2014)
- ATOUT FRANCE - Agence de developpement touristique de la France (2014).

---

`chunkerize`*Multiple code chunks*

---

**Description**

This function creates multiple code chunks from a function and along arguments marked with a star (\*). Each of these special arguments is a list. The nth code chunk will use the nth element of each marked list (recycled if necessary) as argument.

**Usage**

```
chunkerize(FUN, arg.names, arg.values, type = "block", echo = FALSE,  
warning = FALSE, error = FALSE, message = TRUE, fig.width = 4,  
fig.height = 4, fig.align = "center", options = NULL)
```

**Arguments**

<code>FUN</code>	the function to use for the chunks.
<code>arg.names</code>	a character vector giving the argument names of <code>FUN</code> to set.
<code>arg.values</code>	a vector giving the values or object names to assign to each argument given with <code>arg.names</code> (they must match in order). Object names must be backquoted or quoted. Lists names marked with a star (e.g. <code>"*L1"</code> for <code>L1</code> ) indicate their elements will be used sequentially in chunks.
<code>type</code>	the type of chunk to produce. Can be <code>"block"</code> , <code>"inline"</code> or <code>"none"</code> .
<code>echo</code>	logical indicating whether to include R source code in the result.
<code>warning</code>	logical indicating whether to print warnings in the result.
<code>error</code>	logical indicating whether to stop on errors.
<code>message</code>	logical indicating whether to print messages in the result.
<code>fig.width, fig.height</code>	numeric value setting width and height of the plots in inches.
<code>fig.align</code>	character string setting the alignment of the plots. Can be <code>"left"</code> , <code>"right"</code> and <code>"center"</code> .
<code>options</code>	a character string to specify the knitr options. This will overwrite the options set with the other arguments.

**Value**

a character vector of R code chunks which can be evaluated by **knitr**.

---

<code>hotels</code>	<i>French Hotels</i>
---------------------	----------------------

---

**Description**

This dataset gives the number of hotels, number of rooms and capacity for each department of metropolitan France.

**Usage**

```
data(hotels)
```

**Format**

a `SpatialPolygonsDataFrame` with geometries of the 96 french departements (epsg:4326) and 12 variables.

- `DEP.CODE` The code number of each department.
- `DEP.NAME` The name of each department.
- `CHF.NAME` The name of the main (administrative) city of each department.

- REGION.NAME The name of the french region (administrative) of each department.
- N.HOTELS The number of hotels.
- N.5, N.4, N.3, N.2, N.1 The number of hotels for each ranking categories (i.e. stars).
- ROOMS The number of hotel's rooms for each department.
- CAPACITY The total capacity (beds) for each department.

### Source

- Institut National de l'Information Geographique et Forestiere (2014)
- ATOUT FRANCE - Agence de developpement touristique de la France (2014).

---

 layerLegend

*Data layer legend*


---

### Description

This function creates a legend object that can be attached to a data layer.

### Usage

```
layerLegend(style, labels = "", title = NULL, position = "bottomright",
  png = NULL, size = 5, png.width = NULL, png.height = NULL,
  stroke.col = 1, stroke.lwd = 1, stroke.lty = -1, stroke.alpha = 1,
  fill.col = 2, fill.alpha = 0.5, cells.range = c(1, 10),
  cells.col = heat.colors(12), cells.alpha = 1)
```

### Arguments

style	the style of legend to generate. Can be one of "points", "lines", "polygons", "icons", "gradient".
labels	a vector to label each element of the legend.
title	a title which will appear above the legend. If NULL (default), it will inherit the name of the data layer during the compilation of the map. If NA, the legend will appear without title.
position	a character string indicating where should the legend be displayed. This must be one of "topleft", "topright", "bottomleft", "bottomright", "none"
png	character vector giving the paths for the PNG icons. If NULL (default), circles are drawn.
size	a numerical vector giving the size of points (radius in pixels).
png.width	numerical vectors giving the PNG icons dimensions on the map (in pixels).
png.height	numerical vectors giving the PNG icons dimensions on the map (in pixels).
stroke.col	a vector of any of the three kinds of R color specifications to set strokes color.
stroke.lwd	a numerical vector to set strokes width.

<code>stroke.lty</code>	a character vector that defines the strokes dash patterns (See Details).
<code>stroke.alpha</code>	a vector of numeric values in $[0, 1]$ setting strokes opacity.
<code>fill.col</code>	a vector of any of the three kinds of <b>R</b> color specifications to set fill colors.
<code>fill.alpha</code>	a vector of numeric values in $[0, 1]$ setting fill opacity.
<code>cells.range</code>	range of gradient values.
<code>cells.col</code>	a vector of any of the three kinds of <b>R</b> color specifications giving the colors of the gradient.
<code>cells.alpha</code>	a vector of numeric values in $[0, 1]$ setting the gradient opacity.

**Value**

an object `layerlegend` which can be passed to an `spLayer*` function through the `legend` argument.

---

<code>print.splpoints</code>	<i>Printing spl object</i>
------------------------------	----------------------------

---

**Description**

Print spl objects

**Usage**

```
## S3 method for class 'splpoints'
print(x, ...)

## S3 method for class 'splicons'
print(x, ...)

## S3 method for class 'spllines'
print(x, ...)

## S3 method for class 'splpolygons'
print(x, ...)

## S3 method for class 'splgrid'
print(x, ...)
```

**Arguments**

<code>x</code>	an spl object.
<code>...</code>	additional arguments. Not used.



---

rleafmap	<i>rleafmap</i>
----------	-----------------

---

## Description

rleafmap

---

spLayer	<i>Define a Data Layer</i>
---------	----------------------------

---

## Description

Define a new data layer from an object sp.

## Usage

```
spLayer(x, ...)
```

## Arguments

x	a spatial object as defined in the package <b>sp</b> .
...	additional arguments to pass to the function.

## Examples

```
## Not run:
#POINTS
data(velov)
vv <- spLayer(velov, stroke=F, popup=velov$NAME)

#POLYGONS
data(campsites)
gcol <- rev(heat.colors(5))
gcut <- cut(mapdep$N.CAMPSITES, breaks=c(-1, 20, 40, 60, 80, 1000))
cs <- spLayer(campsites, fill.col=as.numeric(gcut))
bm1 <- basemap("mapquest.map")

writeMap(bm1, cs, vv)

## End(Not run)
```

---

spLayer.default      *Define a Vector Data Layer*

---

### Description

Define a Vector Data Layer

### Usage

```
## Default S3 method:
spLayer(x, ...)
```

### Arguments

x                    a spatial object as defined in the package **sp**.  
 ...                  additional arguments to pass to the function.

---

spLayer.SpatialGridDataFrame  
                           *Define a Raster Data Layer*

---

### Description

spLayer.SpatialGridDataFrame defines a new data layer from an object SpatialGridDataFrame.

### Usage

```
## S3 method for class 'SpatialGridDataFrame'
spLayer(x, name = NULL, layer,
        cells.col = heat.colors(12), cells.alpha = 1, legend = NULL, ...)
```

### Arguments

x                    a spatial object (see Details).  
 name                a character string to name the layer.  
 layer                which layer to select?  
 cells.col           a vector of any of the three kinds of **R** color specifications giving a gradient to color the grid.  
 cells.alpha        a vector of numeric values in [0, 1] setting grid opacity.  
 legend             a legend object created with [layerLegend](#).  
 ...                  additional arguments to pass to the function.

---

 spLayer.SpatialLines *Define a Vector Data Layer*


---

**Description**

Define a Vector Data Layer

**Usage**

```
## S3 method for class 'SpatialLines'
spLayer(x, name = NULL, stroke = TRUE,
        stroke.col = 1, stroke.lwd = 1, stroke.lty = -1, stroke.alpha = 1,
        label = NULL, popup = "", popup.rmd = FALSE, legend = NULL, ...)
```

**Arguments**

x	a spatial object (see Details).
name	a character string to name the layer.
stroke	logical. Should a stroke be drawn along lines and polygons?
stroke.col	a vector of any of the three kinds of R color specifications to set strokes color.
stroke.lwd	a numerical vector to set strokes width.
stroke.lty	a character vector that defines the strokes dash patterns (See Details).
stroke.alpha	a vector of numeric values in [0, 1] setting strokes opacity.
label	a reserved argument (in development).
popup	a character vector giving contents for popups. HTML tags are accepted.
popup.rmd	a logical indicating whether the popups should be processed as R Markdown with <b>knitr</b> . Default FALSE.
legend	a legend object created with <a href="#">layerLegend</a> .
...	additional arguments to pass to the function.

---

 spLayer.SpatialPoints *Define a Vector Data Layer*


---

**Description**

- spLayer.SpatialPoints defines a new data layer from an object SpatialPoints or SpatialPointsDataFrame
- spLayer.SpatialLines defines a new data layer from an object SpatialLines or SpatialLinesDataFrame
- spLayer.SpatialPolygons defines a new data layer from an object SpatialPolygons or SpatialPolygonsDataFrame

**Usage**

```
## S3 method for class 'SpatialPoints'
spLayer(x, name = NULL, png = NULL, size = 5,
  png.width = 15, png.height = 15, stroke = TRUE, stroke.col = 1,
  stroke.lwd = 1, stroke.lty = -1, stroke.alpha = 1, fill = TRUE,
  fill.col = 2, fill.alpha = 0.5, label = NULL, popup = "",
  popup.rmd = FALSE, legend = NULL, ...)
```

**Arguments**

x	a spatial object (see Details).
name	a character string to name the layer.
png	character vector giving the paths for the PNG icons. If NULL (default), circles are drawn.
size	a numerical vector giving the size of points (radius in pixels).
png.width, png.height	numerical vectors giving the PNG icons dimensions on the map (in pixels).
stroke	logical. Should a stroke be drawn along lines and polygons?
stroke.col	a vector of any of the three kinds of R color specifications to set strokes color.
stroke.lwd	a numerical vector to set strokes width.
stroke.lty	a character vector that defines the strokes dash patterns (See Details).
stroke.alpha	a vector of numeric values in [0, 1] setting strokes opacity.
fill	logical. Should points and polygons be filled?
fill.col	a vector of any of the three kinds of R color specifications to set fill colors.
fill.alpha	a vector of numeric values in [0, 1] setting fill opacity.
label	a reserved argument (in development).
popup	a character vector giving contents for popups. HTML tags are accepted.
popup.rmd	a logical indicating whether the popups should be processed as R Markdown with <b>knitr</b> . Default FALSE.
legend	a legend object created with <a href="#">layerLegend</a> .
...	additional arguments to pass to the function.

---

spLayer.SpatialPolygons

*Define a Vector Data Layer*


---

**Description**

Define a Vector Data Layer

**Usage**

```
## S3 method for class 'SpatialPolygons'
spLayer(x, name = NULL, stroke = TRUE,
        stroke.col = 1, stroke.lwd = 1, stroke.lty = -1, stroke.alpha = 1,
        fill = TRUE, fill.col = 2, fill.alpha = 0.5, label = NULL,
        popup = "", popup.rmd = FALSE, holes = FALSE, legend = NULL, ...)
```

**Arguments**

x	a spatial object (see Details).
name	a character string to name the layer.
stroke	logical. Should a stroke be drawn along lines and polygons?
stroke.col	a vector of any of the three kinds of R color specifications to set strokes color.
stroke.lwd	a numerical vector to set strokes width.
stroke.lty	a character vector that defines the strokes dash patterns (See Details).
stroke.alpha	a vector of numeric values in [0, 1] setting strokes opacity.
fill	logical. Should points and polygons be filled?
fill.col	a vector of any of the three kinds of R color specifications to set fill colors.
fill.alpha	a vector of numeric values in [0, 1] setting fill opacity.
label	a reserved argument (in development).
popup	a character vector giving contents for popups. HTML tags are accepted.
popup.rmd	a logical indicating whether the popups should be processed as R Markdown with <b>knitr</b> . Default FALSE.
holes	a logical indicating whether to use the hole slots of the SpatialPolygons object.
legend	a legend object created with <a href="#">layerLegend</a> .
...	additional arguments to pass to the function.

---

spLayerControl

*Testing user inputs*


---

**Description**

This function tests arguments validity for the function [spLayer](#).

**Usage**

```
spLayerControl(name, size = 1, legend = legend, stroke = TRUE,
               stroke.col = 1, stroke.lwd = 1, stroke.lty = 1, stroke.alpha = 1,
               fill = TRUE, fill.col = 1, fill.alpha = 1, label = "", popup = "",
               holes = FALSE)
```

**Arguments**

name	a character string to name the layer.
size	a numerical vector giving the size of points (radius in pixels).
legend	a legend object created with <a href="#">layerLegend</a> .
stroke	logical. Should a stroke be drawn along lines and polygons?
stroke.col	a vector of any of the three kinds of R color specifications to set strokes color.
stroke.lwd	a numerical vector to set strokes width.
stroke.lty	a character vector that defines the strokes dash patterns (See Details).
stroke.alpha	a vector of numeric values in [0, 1] setting strokes opacity.
fill	logical. Should points and polygons be filled?
fill.col	a vector of any of the three kinds of R color specifications to set fill colors.
fill.alpha	a vector of numeric values in [0, 1] setting fill opacity.
label	a reserved argument (in development).
popup	a character vector giving contents for popups. HTML tags are accepted.
holes	a logical indicating whether to use the hole slots of the SpatialPolygons object.

---

summary.basemap

*Summary of map elements*


---

**Description**

Get a summary of a map element.

**Usage**

```
## S3 method for class 'basemap'
summary(object, ...)

## S3 method for class 'splpoints'
summary(object, ...)

## S3 method for class 'splicons'
summary(object, ...)

## S3 method for class 'spllines'
summary(object, ...)

## S3 method for class 'splpolygons'
summary(object, ...)

## S3 method for class 'splgrid'
```

```
summary(object, ...)  
  
## S3 method for class 'ui'  
summary(object, ...)
```

### Arguments

`object` a map layer, basemap, spl or ui object.  
`...` additional arguments. Not used.

---

toGeoJSON *Convert a spl object to GeoJSON format*

---

### Description

This function is used internally by [writeMap](#) to convert a given spl object to GeoJSON format.

### Usage

```
toGeoJSON(x, lightjson = F)
```

### Arguments

`x` a spl object.  
`lightjson` logical. Should GeoJSON code size be reduced by suppressing extra whitespace characters and rounding numeric values?

### Value

A character string of GeoJSON formatted code.

### See Also

[writeOGR](#) and its driver "GeoJSON" for GeoJSON export.

---

toJS	<i>Generate Leaflet JS code for a given layer</i>
------	---

---

### Description

This function is used internally by `writeMap` to generate Leaflet JavaScript code for a given layer.

### Usage

```
toJS(x, url = "")
```

### Arguments

x	a spl or basemap object.
url	a character string giving the path for the raster files.

### Value

A character string of JavaScript Code.

---

ui	<i>Options settings for map interface</i>
----	---

---

### Description

Allow the user to choose which interface elements are displayed on the map and their positions.

### Usage

```
ui(zoom = c("topleft", "topright", "bottomleft", "bottomright", "none"),
  layers = c("none", "topright", "topleft", "bottomleft", "bottomright"),
  attrib = c("bottomright", "topleft", "topright", "bottomleft", "none"),
  attrib.text = "")
```

### Arguments

zoom	a character string indicating if and how should the zoom control be displayed. This must be one of "topleft", "topright", "bottomleft", "bottomright", "none"
layers	a character string indicating if and how should the layers control be displayed. This must be one of "topleft", "topright", "bottomleft", "bottomright", "none"
attrib	a character string indicating if and how should the attribution control be displayed. This must be one of "topleft", "topright", "bottomleft", "bottomright", "none"
attrib.text	a character string for additionnal credits. HTML tags are accepted.



**Value**

An object of class ui which can be directly given as interface argument of `writeMap`.

---

uiJS	<i>Generate user interface JS code</i>
------	--

---

**Description**

This function is used internally by `writeMap` to generate JavaScript code related to the user interface.

**Usage**

```
uiJS(interface, ar)
```

**Arguments**

interface	an ui object created with <code>ui</code> .
ar	a list of basemap and spl objects.

---

velov	<i>Velo'v stations</i>
-------	------------------------

---

**Description**

Stations of the bicycle sharing system of the city of Lyon (France).

**Usage**

```
data(velov)
```

**Format**

a `SpatialPointsDataFrame` with the location and name of the 349 Velov stations (epsg:4326).

**Source**

OpenStreetMap (14/04/2014)

---

 writeMap

*Export and display the map*


---

### Description

This function combines all the elements specified by the user and write the corresponding HTML and Javascript code in a local directory.

### Usage

```
writeMap(..., dir = getwd(), prefix = "", width = 700, height = 400,
  setView = NULL, setZoom = NULL, interface = NULL, lightjson = FALSE,
  directView = c("viewer", "browser", "disabled"), leaflet.loc = "online")
```

### Arguments

...	basemap and spl objects to embed into the map
dir	a character string giving the directory path to export the map. Default is the working directory.
prefix	a character string to add a prefix to file names. This allows multiple exportations in the same directory.
width,height	the width and height of the map, in pixels.
setView	a numeric vector of the form <code>c(x, y)</code> setting the initial geographical center of the map.
setZoom	a numeric value setting the initial map zoom.
interface	an ui object created with <code>ui</code> to customize the interface controls.
lightjson	logical. Should GeoJSON files size be reduced by supressing extra whitespace characters and rounding numeric values? Default is FALSE. This is currently not compatible with RMarkdown popups.
directView	a character string indicating if and how should the map be displayed. Default option "viewer" uses (if available) the RStudio HTML viewer to display the map, "browser" opens the map into the web browser and "disabled" disables direct display.
leaflet.loc	a character string specifying the location (directory) of the leaflet library. If set to "online" (default), the library is loaded from the leaflet official CDN and requires an internet connection.

# Index

## \*Topic **datasets**

- [campsites](#), 5
- [hotels](#), 6
- [velov](#), 17

- [basemap](#), 2, 4
- [bmCredit](#), 3
- [bmServer](#), 4
- [bmSource](#), 4, 4

- [campsites](#), 5
- [chunkerize](#), 5

- [hotels](#), 6

- [layerLegend](#), 7, 10–14

- [print.splgrid](#) ([print.splpoints](#)), 8
- [print.splicons](#) ([print.splpoints](#)), 8
- [print.spllines](#) ([print.splpoints](#)), 8
- [print.splpoints](#), 8
- [print.splpolygons](#) ([print.splpoints](#)), 8

- [rleafmap](#), 9
- [rleafmap-package](#) ([rleafmap](#)), 9

- [spLayer](#), 3, 9, 13
- [spLayer.default](#), 10
- [spLayer.SpatialGridDataFrame](#), 10
- [spLayer.SpatialLines](#), 11
- [spLayer.SpatialPoints](#), 11
- [spLayer.SpatialPolygons](#), 12
- [spLayerControl](#), 13
- [summary.basemap](#), 14
- [summary.splgrid](#) ([summary.basemap](#)), 14
- [summary.splicons](#) ([summary.basemap](#)), 14
- [summary.spllines](#) ([summary.basemap](#)), 14
- [summary.splpoints](#) ([summary.basemap](#)), 14
- [summary.splpolygons](#) ([summary.basemap](#)), 14
- [summary.ui](#) ([summary.basemap](#)), 14

- [toGeoJSON](#), 15
- [toJS](#), 16

- [ui](#), 16, 17, 18
- [uiJS](#), 17

- [velov](#), 17

- [writeMap](#), 3, 15–17, 18
- [writeOGR](#), 15