Package ‘leaflet.minicharts’

May 11, 2021

Type Package

Title Mini Charts for Interactive Maps

Version 0.6.2

Description Add and modify small charts on an interactive map created with package 'leaflet'. These charts can be used to represent at same time multiple variables on a single map.

License GPL (>= 2) | file LICENSE

Depends R (>= 2.10)

Imports leaflet (>= 1.1.0), htmltools

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, dplyr, shiny, manipulateWidget, testthat, covr

VignetteBuilder knitr

NeedsCompilation no

Author Veronique Bachelier [aut, cre],
Jalal-Edine ZAWAM [aut],
Benoit Thieurmel [aut],
Francois Guillem [aut],
RTE [cph]

Maintainer Veronique Bachelier <veronique.bachelier@rte-france.com>

Repository CRAN

Date/Publication 2021-05-11 09:20:10 UTC

R topics documented:

`addFlows` *Add or modify flows on a leaflet map*

Description

These functions can be used to represent flows and their evolution on a map created with `leaflet()`. Flows are simply represented by a segment between two points with an arrow at its center that indicates the direction of the flow.

Usage

```
addFlows(
  map,
  lng0,
  lat0,
  lng1,
  lat1,
  color = "blue",
  flow = 1,
  opacity = 1,
  dir = NULL,
  time = NULL,
  popup = popupArgs(labels = "Flow"),
  layerId = NULL,
  timeFormat = NULL,
  initialTime = NULL,
  maxFlow = max(abs(flow)),
  minThickness = 1,
  maxThickness = 20,
  popupOptions = NULL
)

updateFlows(
  map,
  layerId,
  color = NULL,
  flow = NULL,
  opacity = NULL,
  dir = NULL,
  time = NULL,
  popup = NULL,
  timeFormat = NULL,
  initialTime = NULL,
  maxFlow = NULL,
  minThickness = 1,
  maxThickness = 20,
  popupOptions = NULL
)

removeFlows(map, layerId)

clearFlows(map)
```

Arguments

`map`	A leaflet map object created with `leaflet`.
`lng0`	Longitude of the origin of the flow.
`lat0`	Latitude of the origin of the flow.
`lng1`	Longitude of the destination of the flow.
`lat1`	Latitude of the destination of the flow.
`color`	Color of the flow.
`flow`	Value of the flow between the origin and the destination. If argument `dir` is not set, negative values are interpreted as flows from destination to origin.
`opacity`	Opacity of the flow.
`dir`	Direction of the flow. 1 indicates that the flow goes from origin to destination and -1 indicates that it goes from destination to origin. If 0, the arrow is not drawn. If `NULL`, then it is equal to the sign of `weight`.
`time`	A vector with length equal to the number of rows in `chartdata` and containing either numbers representing time indices or dates or datetimes. Each unique value must appear as many times as the others. This parameter can be used when one wants to represent the evolution of some variables on a map.
`popup`	Options that control popup generation.
`layerId`	An ID variable. It is mandatory when one wants to update some chart with `updateMinicharts`.
`timeFormat`	Character string used to format dates and times when argument `time` is a `Date`, `POSIXct` or `POSIXlt` object. See `strptime` for more information.
`initialTime`	This argument can be used to set the initial time step shown when the map is created. It is used only when argument `time` is set.
`maxFlow`	Maximal value a flow could take.
`minThickness`	minimal thickness of the line that represents the flow.
`maxThickness`	maximal thickness of the line that represents the flow.
`popupOptions`	Change default popupOptions (ex : autoClose, maxHeight, closeButton ...) See `popupOptions` for more informations.

Value

The modified leaflet map object.

Examples

```
require(leaflet)

# Toy example
leaflet() %>% addTiles() %>%
  addFlows(0, 0, 1, 1, flow = 10)

# Electric exchanges between France and neighboring countries
data("eco2mixBalance")
bal <- eco2mixBalance
leaflet() %>% addTiles() %>%
  addFlows(
    bal$lng0, bal$lat0, bal$lng1, bal$lat1,
    flow = bal$balance,
    time = bal$month
  )

# popupOptions
data("eco2mixBalance")
bal <- eco2mixBalance
leaflet() %>% addTiles() %>%
  addFlows(
    bal$lng0, bal$lat0, bal$lng1, bal$lat1,
    flow = bal$balance,
    time = bal$month,
    popupOptions = list(closeOnClick = FALSE, autoClose = FALSE)
  )
```

`addMinicharts` *Add or update charts on a leaflet map*

Description

these functions add or update minicharts in a leaflet map at given coordinates: they can be bar charts, pie charts or polar charts where chartdata is encoded either by area or by radius.

Usage

```
addMinicharts(
  map,
  lng,
  lat,
  chartdata = 1,
  time = NULL,
  maxValues = NULL,
  type = "auto",
  fillColor = d3.schemeCategory10[1],
  colorPalette = d3.schemeCategory10,
  width = 30,
  height = 30,
  opacity = 1,
  showLabels = FALSE,
  labelText = NULL,
  labelMinSize = 8,
  labelMaxSize = 24,
  labelStyle = NULL,
  transitionTime = 750,
  popup = popupArgs(),
  layerId = NULL,
  legend = TRUE,
  legendPosition = "topright",
  timeFormat = NULL,
  initialTime = NULL,
  onChange = NULL,
  popupOptions = NULL
)

updateMinicharts(
  map,
  layerId,
  chartdata = NULL,
  time = NULL,
  maxValues = NULL,
  type = NULL,
  fillColor = NULL,
  colorPalette = d3.schemeCategory10,
  width = NULL,
  height = NULL,
  opacity = NULL,
  showLabels = NULL,
  labelText = NULL,
  labelMinSize = NULL,
  labelMaxSize = NULL,
  labelStyle = NULL,
  transitionTime = NULL,
  popup = NULL,
  legend = TRUE,
  legendPosition = NULL,
  timeFormat = NULL,
  initialTime = NULL,
  onChange = NULL,
  popupOptions = NULL
)

removeMinicharts(map, layerId)
```

```
clearMinicharts(map)
```

Arguments

`map`	A leaflet map object created with `leaflet`.
`lng`	Longitude where to place the charts.
`lat`	Latitude where to place the charts.
`chartdata`	A numeric matrix with number of rows equal to the number of elements in `lng` or `lat` and number of column equal to the number of variables to represent. If parameter `time` is set, the number of rows must be equal to the length of `lng` times the number of unique time steps in the data.
`time`	A vector with length equal to the number of rows in `chartdata` and containing either numbers representing time indices or dates or datetimes. Each unique value must appear as many times as the others. This parameter can be used when one wants to represent the evolution of some variables on a map.
`maxValues`	maximal absolute values of the variables to represent. It can be a vector with one value per column of `chartdata` or a single value. Using a single value enforces charts to use a unique scale for all variables. If it is `NULL`, the maximum value of `chartdata` is used.
`type`	Type of chart. Possible values are `"bar"` for bar charts, `"pie"` for pie charts, `"polar-area"` and `"polar-radius"` for polar area charts where the values are represented respectively by the area or the radius of the slices. Finally it can be equal to `"auto"`, the default. In this case, if there is only one variable to represent, the chart will be a single circle, else it is a barchart.
`fillColor`	Used only if data contains only one column. It is the color used to fill the circles.
`colorPalette`	Color palette to use when `chartdata` contains more than one column.
`width`	maximal width of the created elements.
`height`	maximal height of the created elements.
`opacity`	Opacity of the chart.
`showLabels`	Should values be displayed above chart elements.
`labelText`	character vector containing the text content of the charts. Used only if `chartdata` contains only one column.
`labelMinSize`	Minimal height of labels in pixels. When there is not enough space for labels, they are hidden.
`labelMaxSize`	Maximal height of labels in pixels.
`labelStyle`	Character string containing CSS properties to apply to the labels.
`transitionTime`	Duration in milliseconds of the transitions when a property of a chart is updated.
`popup`	Options that control popup generation.
`layerId`	An ID variable. It is mandatory when one wants to update some chart with `updateMinicharts`.
`legend`	If TRUE and if data has column names, then a legend is automatically added to the map.

`legendPosition`	Where should legend be placed?
`timeFormat`	Character string used to format dates and times when argument `time` is a `Date`, `POSIXct` or `POSIXlt` object. See `strptime` for more information.
`initialTime`	This argument can be used to set the initial time step shown when the map is created. It is used only when argument `time` is set.
`onChange`	(For power users who know javascript) A character string containing javascript code that is executed each time a chart is updated. See the details section to understand why and how to use this parameter.
`popupOptions`	Change default popupOptions (ex : autoClose, maxHeight, closeButton ...) See `popupOptions` for more informations.

Details

Since version 0.5, the parameter `onChange` can be used to execute some arbitrary javascript code each time a chart is updated (with `updateMinicharts()` or when time step changes). A typical use case would be to change the color of a polygon added with `addPolygons` based on the data of the chart. It is even possible to create an invisible chart and use it to manage the color and the popup of a polygon. Here is a sample code that do that:

```
leaflet() %>% addTiles() %>%
  addPolygons(data = myPolygons, layerId = myPolygons$myIds) %>%
  addMinicharts(
    mydata$lon, mydata$lat,
    time = mydata$time
    fillColor = mydata$color,
    layerId = mydata$myIds,
    width = 0, height = 0,
    onChange = "
      var s = this._map.layerManager.getLayer("shape", this.layerId);
      s.bindPopup(popup);
      if (opts.fillColor) {
        d3.select(s._path)
        .transition()
        .duration(750)
        .attr("fill", opts.fillColor);
      }"
  )
```

The following objects are available when executing the javascript code:

this The current minichart object. See `https://rte-antares-rpackage.github.io/leaflet.minichart/-_L.Minichart_.html` for more information.

opts The current options passed to the current minichart object.

popup Popup html.

d3 The D3 module.

Here is a toy example

Value

The modified leaflet map object. `addMinicharts` add new minicharts to the map. `updateMinicharts` updates minicharts that have already been added to the map. `removeMinicharts` removes some specific charts from the map and `clearMinicharts` removes all charts from the map and if necessary the legend that has been automatically created.

Examples

```
require(leaflet)
mymap <- leaflet() %>% addTiles() %>% addMinicharts(0, 0, chartdata = 1:3, layerId = "c1")

mymap
mymap %>% updateMinicharts("c1", maxValues = 6)
mymap %>% updateMinicharts("c1", type="pie")

# popupOptions
mymap <- leaflet() %>% addTiles() %>%
  addMinicharts(0, 0, chartdata = 1:3, layerId = "c1", popupOptions = list(closeButton = FALSE))

mymap
mymap %>% updateMinicharts("c1", maxValues = 6, popupOptions = list(closeButton = TRUE))
```

`d3.schemeCategory10` *d3 color palette*

Description

A character vector containing ten colors. These colors are used as the default color palette

Usage

```
d3.schemeCategory10
```

Format

An object of class `character` of length 10.

Author(s)

Francois Guillem

References

https://github.com/d3/d3-scale

`eco2mix` *Electric production, consumption and exchanges of France*

Description

`eco2mix` contains the electric production, consumption and exchanges of France from january 2010 to february 2017 and of 12 french regions from january 2013 to february 2017.

In addition to the total production, the table contains one column for each type of production. The table also contains the latitude and longitude of the center of the regions.

`eco2mixBalance` is an extract of `eco2mix` that contains only exchanges between France and neighbouring countries, in a convenient format to represent flows on a map.

Usage

```
eco2mix

eco2mixBalance
```

Format

An object of class `data.frame` with 686 rows and 22 columns.

An object of class `data.frame` with 430 rows and 7 columns.

Author(s)

Francois Guillem

References

`https://www.rte-france.com/eco2mix`

`popupArgs` *Options for popup generation*

Description

This function simply returns a list of options to control the generation of popups.

Usage

```
popupArgs(
  showTitle = TRUE,
  showValues = TRUE,
  labels = NULL,
  supValues = NULL,
  supLabels = colnames(supValues),
  html = NULL,
  noPopup = FALSE,
  digits = NULL
)
```

Arguments

`showTitle`	If `TRUE` layer id is displayed as title of popups.
`showValues`	If `TRUE`, values are displayed in popups
`labels`	Names of values. If `NULL`, column names of the data bound to a chart are used.
`supValues`	A `data.frame` containing additional values to display in popups.
`supLabels`	Names of the additional values.
`html`	Character vector containing custom html code for popups. You can use this parameter when you are not happy with the default popups.
`noPopup`	If `TRUE`, popups are not created.
`digits`	Max number of decimal digits to display for numeric values. If `NULL`, all digits are displayed.

Value

List containing options for popup generation

`syncWith` *Synchronize multiple maps*

Description

This function can be used when multiple leaflet maps are displayed on the same view (for instance in a shiny application or a Rmarkdown document) and one wants to synchronize their center, zoom and time.

`syncWith()` can also be used with basic leaflet maps to synchronize only their zoom and center.

Usage

```
syncWith(map, groupname)
```

Arguments

`map`	A leaflet map object created with `leaflet`.
`groupname`	Character string. All maps that use the same group name will be synchronized.

Value

The modified leaflet map object.

Examples

```
if (require(manipulateWidget) & require(leaflet)) {

  # Synchronize zoom and center of basic maps.
  basicMap1 <- leaflet() %>% addTiles() %>% syncWith("basicmaps")
  basicMap2 <- leaflet() %>% addTiles() %>% syncWith("basicmaps")
  combineWidgets(basicMap1, basicMap2)

  # Synchronize time step of two maps that represent the evolution of some
  # variable.
  map1 <- leaflet() %>% addTiles() %>%
    addMinicharts(0, 40, chartdata = 1:10, time = 1:10) %>%
    syncWith("maps")
  map2 <- leaflet() %>% addTiles() %>%
    addMinicharts(0, 40, chartdata = 10:1, time = 1:10) %>%
    syncWith("maps")
  combineWidgets(map1, map2)

}
```

Index