Package 'leaftime'

January 26, 2020

Title 'Leaflet-timeline' Plugin for Leaflet

Version 0.2.0

Date 2020-01-26

Maintainer Kent Russell <kent.russell@timelyportfolio.com>

URL https://github.com/timelyportfolio/leaftime

BugReports https://github.com/timelyportfolio/leaftime/issues

Description Use the 'leaflet-timeline' plugin with a leaflet widget to add an interactive slider with play, pause, and step buttons to explore temporal geographic spatial data changes.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Depends R (>= 3.1.0), leaflet (>= 2.0.0)

Imports htmlwidgets, htmltools

Suggests geojsonio

NeedsCompilation no

Author Jonathan Skeate [aut] (leaflet-timeline library, https://github.com/skeate/Leaflet.timeline), Kent Russell [aut, cre] (R interface)

Repository CRAN

Date/Publication 2020-01-26 16:00:02 UTC

R topics documented:

`addTimeline` *Add 'leaflet-timeline' To Leaflet Map*

Description

Add 'leaflet-timeline' To Leaflet Map

Usage

```
addTimeline(
  map = NULL,
  data = NULL,
  group = NULL,
  timelineOpts = timelineOptions(),
  sliderOpts = sliderOptions(),
  width = NULL,
  onchange = NULL
)
```

Arguments

`map`	`htmlwidget` leaflet map to which a timeline will be added.
`data`	`geojson` with data for the timeline. Each feature should have start and end properties so the timeline will know when to show the feature.
`group`	`string` name of the group for the timeline control.
`timelineOpts`	`list` from `timelineOptions`.
`sliderOpts`	`list` from `sliderOptions`.
`width`	valid `CSS` width for the timeline control. If given as a percentage, then `95%` or less is recommended to show within the bounds of the map.
`onchange`	`htmlwidgets::JS` function callback for when the timeline is changed.

Value

`leaflet htmlwidget` with an interactive slider timeline control

See Also

`timelineOptions`,`sliderOptions`

Examples

```
if(interactive()) {

library(leaflet)
library(leaftime)
library(htmltools)

#Build data.frame with 10 obs + 3 cols
power <- data.frame(
  "Latitude" = c(
    33.515556, 38.060556, 47.903056, 49.71, 49.041667, 31.934167,
    54.140586, 54.140586, 48.494444, 48.494444
  ),
  "Longitude" = c(
    129.837222, -77.789444, 7.563056, 8.415278, 9.175, -82.343889,
    13.664422, 13.664422, 17.681944, 17.681944
  ),
  "start" = seq.Date(as.Date("2015-01-01"), by = "day", length.out = 10),
  "end" = seq.Date(as.Date("2015-01-01"), by = "day", length.out = 10) + 1
)

# use geojsonio to convert our data.frame
#  to GeoJSON which timeline expects
power_geo <- geojsonio::geojson_json(power,lat="Latitude",lon="Longitude")

# we can add data in addTimeline
leaflet() %>%
  addTiles() %>%
  setView(44.0665,23.74667,2) %>%
  addTimeline(data = power_geo)

# or we can add data in leaflet()
leaflet(power_geo) %>%
  addTiles() %>%
  setView(44.0665,23.74667,2) %>%
  addTimeline()

# we can control the slider controls through sliderOptions
leaflet(power_geo) %>%
  addTiles() %>%
  setView(44.0665,23.74667,2) %>%
  addTimeline(
    sliderOpts = sliderOptions(
      formatOutput = htmlwidgets::JS(
        "function(date) {return new Date(date).toDateString()}
      "),
      position = "bottomright",
      step = 10,
      duration = 3000,
      showTicks = FALSE
    )
  )
```

```
# we can control the timeline through timelineOptions
#  wondering what should be the default
#  currently timeline uses marker
leaflet(power_geo) %>%
  addTiles() %>%
  setView(44.0665,23.74667,2) %>%
  addTimeline(
    timelineOpts = timelineOptions(
      pointToLayer = htmlwidgets::JS(
"
function(data, latlng) {
  return L.circleMarker(latlng, {
    radius: 3
  })
}
"
      ),
      style = NULL
    )
  )

# change styling manually
leaflet(power_geo) %>%
  addTiles() %>%
  setView(44.0665,23.74667,2) %>%
  addTimeline(
    timelineOpts = timelineOptions(
      pointToLayer = htmlwidgets::JS(
"
function(data, latlng) {
  return L.circleMarker(latlng, {
    radius: 10,
    color: 'black',
    fillColor: 'pink',
    fillOpacity: 1
  })
}
"
      ),
      styleOptions = NULL
    )
  )

# change style with styleOptions helper function
#   this will change style for all points
leaflet(power_geo) %>%
  addTiles() %>%
  setView(44.0665,23.74667,2) %>%
  addTimeline(
    timelineOpts = timelineOptions(
      styleOptions = styleOptions(
        radius = 10,
```

```
        color = "black",
        fillColor = "pink",
        fillOpacity = 1
      )
    )
  )

# to style each point differently based on the data
power_styled <- power
# IE does not like alpha so strip colors of alpha hex
power_styled$color <- substr(topo.colors(6)[ceiling(runif(nrow(power),0,6))],1,7)
power_styled$radius <- seq_len(nrow(power_styled)) # ceiling(runif(nrow(power),3,10))

leaflet(geojsonio::geojson_json(power_styled)) %>%
  addTiles() %>%
  setView(44.0665,23.74667,2) %>%
  # addCircleMarkers(
  #   data = power_styled, lat = ~Latitude, lng = ~Longitude, radius = 11
  # ) %>%
  addTimeline(
    timelineOpts = timelineOptions(
      styleOptions = NULL, # make sure default style does not override
      pointToLayer = htmlwidgets::JS(
"
function(data, latlng) {
  return L.circleMarker(
    latlng,
    {
      radius: +data.properties.radius,
      color: data.properties.color,
      fillColor: data.properties.color,
      fillOpacity: 1
    }
  );
}
"
      )
    )
  )



# we can use onchange to handle timeline change event
leaflet(power_geo) %>%
  addTiles() %>%
  setView(44.0665,23.74667,2) %>%
  addTimeline(
    onchange = htmlwidgets::JS("function(e) {console.log(e, arguments)}")
  )


leaflet(power_geo, elementId = "leaflet-wide-timeline") %>%
  addTiles() %>%
```

```
  setView(44.0665,23.74667,2) %>%
  addTimeline(
    width = "96%"
  )

}
```

`leaftimeDependency` *'Leaflet.timeline' Dependencies*

Description

'Leaflet.timeline' Dependencies

Usage

```
leaftimeDependency()
```

Value

`htmltools::htmlDependency`

`sliderOptions` *Timeline Slider Options Helper*

Description

Timeline Slider Options Helper

Usage

```
sliderOptions(
  start = NULL,
  end = NULL,
  position = NULL,
  formatOutput = formatOutputFun(),
  enablePlayback = NULL,
  enableKeyboardControls = NULL,
  steps = NULL,
  duration = NULL,
  waitToUpdateMap = NULL,
  showTicks = NULL
)
```

Arguments

`start`	`number` that will be the starting value of the slider.
`end`	`number` that will be the ending value of the slider.
`position`	`string` that will be the position of the timeline. See position options.
`formatOutput`	`htmlwidgets::JS` function that outputs the date as a string in the timeline.
`enablePlayback`	`logical` to show playback controls.
`enableKeyboardControls`	`logical` to allow playback to be controlled by the keyboard.
`steps`	`number` for how many steps in the timeline.
`duration`	`number` for the minimum time in milliseconds of the length of playback.
`waitToUpdateMap`	`logical` to wait until user is finished before redrawing.
`showTicks`	`logical` to show ticks on the slider.

Value

`list` of options to customize the timeline slider

See Also

`addTimeline`

`styleOptions` *Timeline Style Options Helper*

Description

Timeline Style Options Helper

Usage

```
styleOptions(
  radius = 3,
  color = NULL,
  stroke = TRUE,
  fill = TRUE,
  fillColor = NULL,
  fillOpacity = NULL
)
```

Arguments

radius	number to specify radius of drawn circle.
color, stroke, fillColor	valid CSS color the circle. fill and/or stroke will override color.
fill	logical to determine if drawn will be filled with color.
fillOpacity	number between 0 and 1 to set opacity of the drawn circle.

Value

list with options to style the timeline

See Also

addTimeline

---

timelineOptions *Timeline Options Helper*

Description

Timeline Options Helper

Usage

```
timelineOptions(
  getInterval = NULL,
  pointToLayer = pointToLayerFun(),
  styleOptions = leaftime::styleOptions(),
  drawOnSetTime = NULL
)
```

Arguments

getInterval	htmlwidgets::JS function that returns an object with start and end properties to specify the start and end of the timeline range. See getInterval.
pointToLayer	htmlwidgets::JS function that determines what is drawn on the map. By default, a circle marker will be drawn. See pointToLayer.
styleOptions	list from styleOptions.
drawOnSetTime	logical to draw when time is set. Default is TRUE. See drawOnSetTime.

Value

list with options to customize the timeline

See Also

addTimeline

Index