

# Package: leaftime (via r-universe)

August 23, 2024

**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

**Repository** <https://timelyportfolio.r-universe.dev>

**RemoteUrl** <https://github.com/timelyportfolio/leaftime>

**RemoteRef** master

**RemoteSha** 5896db939cfbb20078dd85947a879d739e2eb656

## Contents

addTimeline . . . . .	2
leaftimeDependency . . . . .	6
sliderOptions . . . . .	6
styleOptions . . . . .	7
timelineOptions . . . . .	8

<b>Index</b>	<b>9</b>
--------------	----------

---

`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

<code>map</code>	htmlwidget leaflet map to which a timeline will be added.
<code>data</code>	geojson with data for the timeline. Each feature should have start and end properties so the timeline will know when to show the feature.
<code>group</code>	string name of the group for the timeline control.
<code>timelineOpts</code>	list from <a href="#">timelineOptions</a> .
<code>sliderOpts</code>	list from <a href="#">sliderOptions</a> .
<code>width</code>	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.
<code>onchange</code>	<code>htmlwidgets::JS</code> 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(leaflet.time)  
  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	<i>'Leaflet.timeline' Dependencies</i>
--------------------	--

---

**Description**

'Leaflet.timeline' Dependencies

**Usage**

```
leaftimeDependency()
```

**Value**

```
htmltools::htmlDependency
```

---

sliderOptions	<i>Timeline Slider Options Helper</i>
---------------	---------------------------------------

---

**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 <a href="#">position options</a> .
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	<i>Timeline Style Options Helper</i>
--------------	--------------------------------------

---

**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**

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

**Value**

list with options to customize the timeline

**See Also**[addTimeline](#)



# Index

`addTimeline`, [2](#), [7](#), [8](#)

`leafTimeDependency`, [6](#)

`sliderOptions`, [2](#), [6](#)

`styleOptions`, [7](#), [8](#)

`timelineOptions`, [2](#), [8](#)