Package 'datamaps'

May 14, 2018

Title Create Interactive Web Maps with the 'JavaScript Datamaps' Library

Version 0.0.3

Description
Easily create interactive choropleth maps then add bubbles and arcs by coordinates or region name. These maps can be used directly from the console, from 'RStudio', in 'Shiny' apps and 'R Markdown' documents. 'Shiny'
proxies allow to interactively add arcs and bubbles, change choropleth values, or change labels.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports htmlwidgets, magrittr

Suggests shiny, knitr, rmarkdown, RColorBrewer, countrycode, dplyr

RoxygenNote 6.0.1

URL http://datamaps.john-coene.com

BugReports https://github.com/JohnCoene/datamaps/issues

VignetteBuilder knitr

NeedsCompilation no

Author John Coene [aut, cre]

Maintainer John Coene <jcoenep@gmail.com>

Repository CRAN

Date/Publication 2018-05-14 20:20:29 UTC

R topics documented:

`add_arcs` *Add arcs*

Description

Add arcs by coordinates.

Usage

```
add_arcs(p, origin.lon, origin.lat, destination.lon, destination.lat, ...)
```

Arguments

`p`	a datamaps object.
`origin.lon, origin.lat`	origin coordinates.
`destination.lon, destination.lat`	destination coordinates.
`...`	any other arguments to use as options.

Examples

```
states <- data.frame(ori.lon = c(-97.03720, -87.90446),
    ori.lat = c(32.89595, 41.97960),
    des.lon = c(-106.60919, -97.66987),
    des.lat = c(35.04022, 30.19453),
    strokeColor = c("blue", "red"),
    arcSharpness = c(2, 1))
```

```
states %>%
    datamaps(scope = "USA", default = "lightgray") %>%
    add_arcs(ori.lon, ori.lat, des.lon, des.lat, strokeColor)
```

`add_arcs_name` *Add arcs*

Description

Add arcs by name of country of state.

Usage

```
add_arcs_name(p, origin, destination, ...)
```

Arguments

`p`	a datamaps object.
`origin, destination`	edges.
`...`	any other arguments to use as options.

Examples

```
data <- data.frame(origin = c("USA", "FRA", "CHN", "RUS", "COG", "DZA"),
    target = c("FRA", "RUS", "BEL", "CAF", "VEN", "SWZ"),
    greatArc = rep(c(TRUE, FALSE), 3),
    arcSharpness = 2)

data %>%
    datamaps() %>%
    add_arcs_name(origin, target, greatArc, arcSharpness)

# US states
states <- data.frame(origin = c("AR", "NY", "CA", "IL", "CO", "MT",
                                "TX", "WA", "TN", "MT"),
    target = c("OR", "SD", "WI", "TX", "LA", "AZ", "FL", "MI", "HI",
               "OK"),
    strokeWidth = runif(10, 1, 9),
    strokeColor = colorRampPalette(c("red", "blue"))(10))

states %>%
    datamaps(scope = "USA", default = "lightgray") %>%
    add_arcs_name(origin, target, strokeWidth, strokeColor)
```

add_bubbles *Add bubbles*

Description

Add bubbles to the map.

Usage

```
add_bubbles(p, lon, lat, radius, color, name, ..., colors = c("#FFEDA0",
  "#FEB24C", "#F03B20"))
```

Arguments

p	a datamaps object.
lon, lat	coordinates of bubbles.
radius	radius of bubbles.
color	color of bubbles.
name	name of bubbles.
...	any other variable to use in tooltip.
colors	color palette.

Examples

```
coords <- data.frame(city = c("London", "New York", "Beijing", "Sydney"),
                     lon = c(-0.1167218, -73.98002, 116.3883, 151.18518),
                     lat = c(51.49999, 40.74998, 39.92889, -33.92001),
                     values = runif(4, 5, 17))

coords %>%
    datamaps() %>%
    add_bubbles(lon, lat, values * 2, values, city)

data <- data.frame(name = c("USA", "FRA", "CHN", "RUS", "COG", "DZA"),
    color = round(runif(6, 1, 10)))

data %>%
    datamaps(default = "lightgray") %>%
    add_choropleth(name, color) %>%
    add_data(coords) %>%
    add_bubbles(lon, lat, values * 2, values, city, colors = c("red", "blue"))
```

add_choropleth *Add choropleth*

Description

Add choropleth data.

Usage

```
add_choropleth(p, locations, color, ..., colors = c("#FFEDA0", "#FEB24C",
  "#F03B20"))
```

Arguments

p	a datamaps object.
locations	column containing location names as iso3c.
color	column containing color of each location.
...	any other variable to use for tooltip.
colors	color palette.

Examples

```
data <- data.frame(name = c("USA", "FRA", "CHN", "RUS", "COG", "DZA"),
    color = round(runif(6, 1, 10)))

data %>%
    datamaps() %>%
    add_choropleth(name, color, colors = c("skyblue", "yellow", "orangered"))

# categorical colors
cat <- data.frame(name = c("USA", "BRA", "COL", "CAN", "ARG", "CHL"),
    col = rep(c("Yes", "No"), 6))

cat %>%
    datamaps(projection = "orthographic") %>%
    add_choropleth(name, col, colors = c("red", "blue"))

# US states
states <- data.frame(st = c("AR", "NY", "CA", "IL", "CO", "MT", "TX"),
    val = c(10, 5, 3, 8, 6, 7, 2))

states %>%
    datamaps(scope = "usa", default = "lightgray") %>%
    add_choropleth(st, val) %>%
    add_labels()
```

add_data *Add data*

Description

Add a dataset.

Usage

```
add_data(p, data)
```

Arguments

p	a datamaps object.
data	data.frame.

Examples

```
coords <- data.frame(city = c("London", "New York", "Beijing", "Sydney"),
                     lon = c(-0.1167218, -73.98002, 116.3883, 151.18518),
                     lat = c(51.49999, 40.74998, 39.92889, -33.92001),
                     values = c(11, 23, 29 , 42))

data <- data.frame(name = c("USA", "FRA", "CHN", "RUS", "COG", "DZA",
                            "BRA", "AFG"),
    color = round(runif(8, 1, 10)))

edges <- data.frame(origin = c("USA", "FRA", "BGD", "ETH", "KHM", "GRD",
                               "FJI", "GNB", "AUT", "YEM"),
    target = c("BRA", "USA", "URY", "ZAF", "SAU", "SVK", "RWA", "SWE",
               "TUV", "ZWE"),
    strokeColor = rep(c("gray", "black"), 5))

data %>%
    datamaps(default = "lightgray") %>%
    add_choropleth(name, color) %>%
    add_data(coords) %>%
    add_bubbles(lon, lat, values, values, city, colors = c("skyblue", "darkblue")) %>%
    add_data(edges) %>%
    add_arcs_name(origin, target, strokeColor)
```

add_graticule *Add graticule*

Description

Add graticule.

Usage

```
add_graticule(p)
```

Arguments

p	a datamaps object.

Examples

```
datamaps(projection = "orthographic") %>%
  add_graticule()
```

add_icons *Add icons*

Description

Add icons at coordinates.

Usage

```
add_icons(p, lon, lat, ...)

icons_options(p, class = "datamaps-icon", ...)
```

Arguments

p	a datamaps object.
lon, lat	coordinates.
...	any other parameter.
class	a valid CSS class.

See Also

Plugin documentation

Examples

```
coords <- data.frame(
  city = c("London", "New York", "Beijing", "Sydney"),
  lon = c(-0.1167218, -73.98002, 116.3883, 151.18518),
  lat = c(51.49999, 40.74998, 39.92889, -33.92001)
)

coords %>%
  datamaps() %>%
  add_icons(lon, lat)
```

`add_labels` *Add labels*

Description

Add data labels.

Usage

```
add_labels(p, label.color = "#000", line.width = 1, font.size = 10,
  font.family = "Verdana", ...)
```

Arguments

`p`	a datamaps object.
`label.color`	label color.
`line.width`	width of line.
`font.size`	font size.
`font.family`	font family.
`...`	any other parameter.

Examples

```
states <- data.frame(st = c("AR", "NY", "CA", "IL", "CO", "MT", "TX"),
    val = c(10, 5, 3, 8, 6, 7, 2))

states %>%
    datamaps("usa") %>%
    add_choropleth(st, val) %>%
    add_labels(label.color = "blue")
```

add_legend *Add legend*

Description

Add a legend to the map.

Usage

```
add_legend(p)
```

Arguments

p	a datamaps object

Examples

```
data <- data.frame(name = c("USA", "FRA", "CHN", "RUS", "COG", "DZA"),
    values = c("N. America", "EU", "Asia", "EU", "Africa", "Africa"))

data %>%
    datamaps() %>%
    add_choropleth(name, values, colors = c("skyblue", "yellow", "orangered")) %>%
    add_legend()
```

add_markers *Add markers*

Description

Add custom markers at coordinates.

Usage

```
add_markers(p, lon, lat, ...)

markers_options(p, ...)
```

Arguments

p	a datamaps object.
lon, lat	coordinates.
...	any other parameter.

Note

Icons may not show in RStudio viewer, open in browser.

See Also

Plugin documentation

Examples

```
coords <- data.frame(
  city = c("London", "New York", "Beijing", "Sydney"),
  lon = c(-0.1167218, -73.98002, 116.3883, 151.18518),
  lat = c(51.49999, 40.74998, 39.92889, -33.92001),
  radius = runif(4, 5, 17)
)

icon_url <- paste0(
  "https://pbs.twimg.com/profile_images/",
  "927645314630193158/ufoYTbbi_400x400.jpg"
)

coords %>%
  datamaps() %>%
  markers_options(
    icon = list(
      url = icon_url,
      width = 20, height = 20
    ),
    fillOpacity = 1
  ) %>%
  add_markers(lon, lat)
```

`config_arcs` *Configure arcs*

Description

Define options of the arcs.

Usage

```
config_arcs(p, stroke.color = "#DD1C77", stroke.width = 1,
  arc.sharpness = 1, animation.speed = 600, popup.on.hover = FALSE, ...)
```

Arguments

p	a datamaps object.
stroke.color	arc colors.
stroke.width	arc width.
arc.sharpness	arc sharpness.
animation.speed	arc draw speed in milliseconds.
popup.on.hover	whether to show tooltip.
...	any additional options.

Examples

```
edges <- data.frame(origin = c("USA", "FRA", "BGD", "ETH", "KHM",
                               "GRD", "FJI", "GNB", "AUT", "YEM"),
    target = c("BRA", "USA", "URY", "ZAF", "SAU", "SVK", "RWA", "SWE",
               "TUV", "ZWE"))

edges %>%
    datamaps() %>%
    add_arcs_name(origin, target) %>%
    config_arcs(stroke.color = "blue", stroke.width = 2, arc.sharpness = 1.5,
                animation.speed = 1000)
```

---

config_bubbles *Configure bubbles*

Description

Define options of the bubbles.

Usage

```
config_bubbles(p, popup.on.hover = TRUE, highlight.on.hover = TRUE,
  fill.opacity = 0.75, animate = TRUE, border.width = 1,
  border.opacity = 1, border.color = "#FDFDFD",
  highlight.fill.color = "#FC8D59", highlight.border.opacity = 1,
  highlight.border.color = "rgba(250, 15, 160, 0.2)",
  highlight.fill.opacity = 0.85, highlight.border.width = 2,
  exit.delay = 100, ...)
```

Arguments

p	a datamaps object.
popup.on.hover	whether to show popover.
highlight.on.hover	whether to enable popover.
fill.opacity	opacity of bubbles.
animate	Wether to animate bubbles.
border.width	width of bubbles.
border.opacity	opacity of bubbles' border.
border.color	color of bubbles' border.
highlight.fill.color	bubbles fill color on hover.
highlight.border.opacity	bubbles opacity on hover.
highlight.border.color	bubble's border opacity on hover.
highlight.fill.opacity	bubble's opacity on hover.
highlight.border.width	bubble's width on hover.
exit.delay	highlight delay.
...	any other parameter.

Examples

```
coords <- data.frame(city = c("London", "New York", "Beijing", "Sydney"),
                     lon = c(-0.1167218, -73.98002, 116.3883, 151.18518),
                     lat = c(51.49999, 40.74998, 39.92889, -33.92001),
                     values = runif(4, 3, 20))

coords %>%
    datamaps(default = "lightgray") %>%
    add_bubbles(lon, lat, values * 2, values, city) %>%
    config_bubbles(highlight.border.color = "rgba(0, 0, 0, 0.2)",
                   fill.opacity = 0.6,
                   border.width = 0.7,
                   highlight.border.width = 5,
                   highlight.fill.color = "green")
```

config_geo *Configure map*

Description

Define options of the map.

Usage

```
config_geo(p, popup.on.hover = TRUE, highlight.on.hover = TRUE,
  hide.antarctica = TRUE, hide.hawaii.and.alaska = FALSE,
  border.width = 1, border.opacity = 1, border.color = "#FDFDFD",
  highlight.fill.color = "#FC8D59", highlight.border.opacity = 1,
  highlight.border.color = "rgba(250, 15, 160, 0.2)",
  highlight.fill.opacity = 0.85, highlight.border.width = 2,
  data.url = NULL, ...)
```

Arguments

p	a datamaps object.
popup.on.hover	whether to show popover.
highlight.on.hover	whether to enable popover.
hide.antarctica	whether to hide Antarctica.
hide.hawaii.and.alaska	whether to hide hide Hawaii and Alaska.
border.width	country border width.
border.opacity	Opacity of country borders.
border.color	color of country borders.
highlight.fill.color	bubbles fill color on hover.
highlight.border.opacity	bubbles opacity on hover.
highlight.border.color	bubble's border opacity on hover.
highlight.fill.opacity	bubble's opacity on hover.
highlight.border.width	bubble's width on hover.
data.url	topo.json data url.
...	any other parameter.

Examples

```
data <- data.frame(name = c("USA", "FRA", "CHN", "RUS", "COG", "DZA"),
    values = c("N. America", "EU", "Asia", "EU", "Africa", "Africa"),
    letters = LETTERS[1:6])

data %>%
    datamaps(default = "lightgray") %>%
    add_choropleth(name, values) %>%
    config_geo(hide.antarctica = FALSE,
               border.width = 2,
               border.opacity = 0.6,
               border.color = "gray",
               highlight.border.color = "green",
               highlight.fill.color = "lightgreen")
```

`datamaps` *Initiate map*

Description

Setup a datamaps chart.

Usage

```
datamaps(data, scope = "world", default = "#ABDDA4",
  projection = "equirectangular", responsive = TRUE, width = "100%",
  height = "100%", elementId = NULL)
```

Arguments

`data`	data.frame.
`scope`	map scope.
`default`	default color for missing values.
`projection`	map projection.
`responsive`	whether for map to be responsive.
`width, height`	Must be a valid CSS unit (like `'100%'`, `'400px'`, `'auto'`) or a number, which will be coerced to a string and have `'px'` appended.
`elementId`	DOM id.

Examples

```
datamaps(projection = "orthographic") %>%
  add_graticule()
```

datamaps-shiny *Shiny bindings for datamaps*

Description

Output and render functions for using datamaps within Shiny applications and interactive Rmd documents.

Usage

```
datamapsOutput(outputId, width = "100%", height = "400px")

renderDatamaps(expr, env = parent.frame(), quoted = FALSE)

datamapsProxy(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a datamaps
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.
id	id of map.
session	shiny session.

delete_map *Remove map*

Description

Remove the map

Usage

```
delete_map(proxy)
```

Arguments

proxy	a proxy as returned by datamapsProxy.

Examples

```
## Not run:
library(shiny)

ui <- fluidPage(
  actionButton(
    "delete",
    "Delete map"
  ),
  datamapsOutput("map")
)

server <- function(input, output){
  output$map <- renderDatamaps({
    datamaps()
  })
}

shinyApp(ui, server)

## End(Not run)
```

`set_projection` *Set projection*

Description

Set projection

Usage

```
set_projection(p, fun = htmlwidgets::JS())
```

Arguments

p	a datamaps object.
fun	a JavaScript function.

Note

Does not work in RStudio viewer, open in browser.

See Also

https://github.com/Anujarya300/bubble_maps

Examples

```
topo <- paste0("https://rawgit.com/Anujarya300/bubble_maps/",
  "master/data/geography-data/india.topo.json")

data <- data.frame(state = c("JH", "MH"), value = c(55, 28))

data %>%
  datamaps(scope = "india") %>%
  add_choropleth(state, value) %>%
  config_geo(data.url = topo) %>%
  set_projection(htmlwidgets::JS('
  function (element) {
    var projection = d3.geo.mercator()
    .center([78.9629, 23.5937])
    .scale(1000);
    var path = d3.geo.path().projection(projection);
    return { path: path, projection: projection };
  }
  ')
  )
```

`update_arcs` *Dynamically update arcs*

Description

Dynamically update arcs with Shiny.

Usage

```
update_arcs(proxy, origin.lon, origin.lat, destination.lon, destination.lat,
  ...)

update_arcs_name(proxy, origin, destination, ...)
```

Arguments

`proxy`	a proxy as returned by `datamapsProxy`.
`origin.lon`	origin coordinates.
`origin.lat`	origin coordinates.
`destination.lon`	destination coordinates.
`destination.lat`	destination coordinates.
`...`	any other arguments to use as options.
`origin`	edges.
`destination`	edges.

Examples

```
## Not run:
library(shiny)

ui <- fluidPage(

  textInput(
    "from",
    "Origin",
    value = "USA"
  ),
  textInput(
    "to",
    "Destination",
    value = "RUS"
  ),
  actionButton(
    "submit",
    "Draw arc"
  ),
  datamapsOutput("map")
)

server <- function(input, output){

  arc <- reactive({
    data.frame(from = input$from, to = input$to)
  })

 output$map <- renderDatamaps({
   datamaps()
 })

 observeEvent(input$submit, {
   datamapsProxy("map") %>%
     add_data(arc()) %>%
     update_arcs_name(from, to)
 })

}

shinyApp(ui, server)

## End(Not run)
```

---

update_bubbles *Dynamically add bubbles*

Description

Dynamically add bubble using Shiny.

Usage

```
update_bubbles(proxy, lon, lat, radius, color, name, ...)
```

Arguments

proxy	a proxy as returned by datamapsProxy.
lon, lat	coordinates of bubbles.
radius	radius of bubbles.
color	color of bubbles.
name	name of bubbles.
...	any other variable to use in tooltip.

Examples

```
## Not run:
library(shiny)

ui <- fluidPage(
 numericInput(
   "lon",
   "Longitude",
   value = 50
 ),
 numericInput(
   "lat",
   "Latitude",
   value = 50
 ),
 textInput(
   "city",
   "City",
   value = "City"
 ),
 sliderInput(
   "value",
   "Value",
   min = 1,
   max = 4,
   step = 1,
   value = 3
 ),
 actionButton(
   "sub",
   "Submit"
 ),
 datamapsOutput("map")
```

```
)

server <- function(input, output){

  coords <- data.frame(city = c("London", "New York", "Beijing", "Sydney"),
                       lon = c(-0.1167218, -73.98002, 116.3883, 151.18518),
                       lat = c(51.49999, 40.74998, 39.92889, -33.92001),
                       values = 1:4)

  update <- reactive({
   df <- data.frame(city = input$city, lon = input$lon, lat = input$lat, values = input$value)
    rbind.data.frame(coords, df)
  })

  output$map <- renderDatamaps({
    coords %>%
      datamaps() %>%
      add_bubbles(lon, lat, values * 2, values, city)
  })

  observeEvent(input$sub, {
    datamapsProxy("map") %>%
      add_data(update()) %>% # pass updated data
      update_bubbles(lon, lat, values * 2, values, city) # update
  })

}

shinyApp(ui, server)

## End(Not run)
```

`update_choropleth` *Dynamically add bubbles*

Description

Dynamically add bubbles using Shiny.

Usage

```
update_choropleth(proxy, locations, color, reset = FALSE, ...)
```

Arguments

`proxy`	a proxy as returned by `datamapsProxy`.
`locations`	column containing location names as `iso3c`.
`color`	column containing color of each `location`.

reset	reset previous changes to default color from datamaps.
...	any other variable to use for tooltip.

Examples

```
## Not run:
library(shiny)

ui <- fluidPage(
  selectInput(
    "countrySelect",
    "Select Country",
    choices = c("USA", "FRA", "CHN", "RUS", "COG", "DZA", "BRA", "IND")
  ),
  sliderInput(
    "value",
    "Value",
    min = 1,
    max = 10,
    value = 5
  ),
  actionButton("update", "Update"),
  datamapsOutput("map")
)

server <- function(input, output){

  data <- data.frame(name = c("USA", "FRA", "CHN", "RUS", "COG", "DZA", "BRA", "IND", "ALG", "AFG"),
                     color = 1:10)

 updated_data <- reactive({
   data.frame(name = input$countrySelect, value = input$value)
 })

  output$map <- renderDatamaps({
    data %>%
      datamaps() %>%
      add_choropleth(name, color)
  })

  observeEvent(input$update, {
    datamapsProxy("map") %>%
      add_data(updated_data()) %>% # pass updated data
      update_choropleth(name, value, TRUE) # update
  })
}

shinyApp(ui, server)

## End(Not run)
```

update_labels *Dynamically update labels*

Description

Dynamically update labels using Shiny

Usage

```
update_labels(proxy, label.color = "#000", line.width = 1, font.size = 10,
  font.family = "Verdana", ...)
```

Arguments

proxy	a proxy as returned by datamapsProxy.
label.color	color of label.
line.width	with of line.
font.size	size of font label.
font.family	family of font label.
...	any other option.

Examples

```
## Not run:
library(shiny)

ui <- fluidPage(
  actionButton(
    "update",
    "update labels"
  ),
  datamapsOutput("map")
)

server <- function(input, output){
  states <- data.frame(st = c("AR", "NY", "CA", "IL", "CO", "MT", "TX"),
                       val = c(10, 5, 3, 8, 6, 7, 2))

  output$map <- renderDatamaps({
    states %>%
      datamaps(scope = "usa", default = "lightgray") %>%
      add_choropleth(st, val) %>%
      add_labels()
  })

  observeEvent(input$update, {
    datamapsProxy("map") %>%
      update_labels(sample(c("blue", "red", "orange", "green", "white"), 1)) # update
```

```
  })
}

shinyApp(ui, server)

## End(Not run)
```

---

update_legend *Dynamically update legend*

Description

Dynamically update legend using Shiny

Usage

```
update_legend(proxy)
```

Arguments

proxy	a proxy as returned by datamapsProxy.

Examples

```
## Not run:
library(shiny)

ui <- fluidPage(
  actionButton(
    "show",
    "Show legend"
  ),
  datamapsOutput("map")
)

server <- function(input, output){
  states <- data.frame(st = c("AR", "NY", "CA", "IL", "CO", "MT", "TX"),
                       val = c(10, 5, 3, 8, 6, 7, 2))

  output$map <- renderDatamaps({
    states %>%
      datamaps(scope = "usa", default = "lightgray") %>%
      add_choropleth(st, val)
  })

  observeEvent(input$update, {
    datamapsProxy("map") %>%
      update_legend() # update
  })
```

```
}

shinyApp(ui, server)

## End(Not run)
```

Index