Package 'leafletZH'

November 10, 2024

Type Package

Title Chinese Leaflet Map Relate Operation

Version 0.1.1

Maintainer Damonsoul <chenmaowei96@gmail.com>

Description Provides 'sf' data for Chinese provinces and cities,
methods for plotting shape maps of Chinese provinces and cities,
Convert Coordinates Between Different Systems,
and a layer for 'leaflet' with Gaode tiles.
It is designed to facilitate geographical data visualization in China.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports geojsonsf, geosphere, grDevices, htmltools, htmlwidgets,
leaflet, leaflet.extras, purrr, Rcpp, scales, sf, stringr

RoxygenNote 7.3.2

Depends R (>= 4.0.0)

LinkingTo Rcpp, RcppArmadillo

URL https://damonsoul.github.io/leafletZH/

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

Author Damonsoul [aut, cre]

Repository CRAN

Date/Publication 2024-11-10 16:40:05 UTC

Contents

`addAreaPolygons` *Add Area Polygons to a Map*

Description

This function adds a polygon area to a given map using the specified latitude and longitude coordinates.

Usage

```
addAreaPolygons(map, longitude, latitude, coordinate = "WGS-84")
```

Arguments

`map`	The map object to which the polygons will be added.
`longitude`	A vector of longitudes.
`latitude`	A vector of latitudes.
`coordinate`	A string indicating the coordinate system of the input data. Options are "WGS-84", "GCJ-02", "BD-09".

Value

The updated map object with added polygons.

Examples

```
library(leaflet)
m <- leaflet() %>% addTilesAmap()
m <- addAreaPolygons(m,
  longitude = c(121.0, 122.1, 121.2, 122.15, 121.2),
  latitude =  c(31.1, 31.919, 31.917, 31.15, 31.12), coordinate = "WGS-84"
)
m
```

addCityShape *Adds a choropleth map layer for citys with additional customization options.*

Description

Adds a choropleth map layer for citys with additional customization options.

Usage

```
addCityShape(
  map,
  data,
  adcode = NULL,
  layerId = NULL,
  group = NULL,
  valueProperty = NULL,
  labelProperty = NULL,
  labelOptions = leaflet::labelOptions(),
  popupProps = NULL,
  popupOptions = leaflet::popupOptions(),
  scale = c("white", "red"),
  steps = 5,
  mode = "q",
  channelMode = c("rgb", "lab", "hsl", "lch"),
  padding = NULL,
  correctLightness = FALSE,
  bezierInterpolate = FALSE,
  colors = NULL,
  stroke = TRUE,
  color = "#ffffff",
  weight = 1,
  opacity = 0.5,
  fillOpacity = 0.7,
  dashArray = NULL,
  smoothFactor = 1,
  noClip = FALSE,
  pathOptions = leaflet::pathOptions(),
 highlightOptions = leaflet::highlightOptions(weight = 2, color = "#000000", fillOpacity
    = 1, opacity = 1, bringToFront = TRUE, sendToBack = TRUE),
  legendOptions = NULL,
  ...
)
```

Arguments

map	The leaflet map object to add the layer to.

data	A data frame containing the data to be visualized.
adcode	China administrative division code
layerId	An optional string to identify the layer.
group	An optional string for grouping data.
valueProperty	The property in the geojson data that corresponds to the value to be mapped.
labelProperty	The property in the geojson data that will be used for labels.
labelOptions	Options for labels, defaults to leaflet's labelOptions.
popupProps	A named vector of properties to display in the popup.
popupOptions	Options for popups, defaults to leaflet's popupOptions.
scale	A vector of colors to use for the scale of the choropleth map.
steps	The number of steps for the color scale.
mode	The mode for the color scale, can be "q" for quantile, "e" for equal interval, etc.
channelMode	The color channel mode, can be "rgb", "lab", "hsl", or "lch".
padding	Optional padding for the choropleth layer.
correctLightness	A logical value to correct lightness for color scales.
bezierInterpolate	Whether to use bezier interpolation for the lines.
colors	An optional vector of colors to override the default color scale.
stroke	Whether to draw the stroke along the paths.
color	The color for the paths, defaults to white.
weight	The weight for the paths.
opacity	The opacity for the paths.
fillOpacity	The fill opacity for the paths.
dashArray	An optional array to create dashed lines.
smoothFactor	A factor to smooth the factor for the paths.
noClip	Whether to disable clipping of the paths.
pathOptions	Additional options for the paths, defaults to leaflet's pathOptions.
highlightOptions	Options for highlighting features, defaults to leaflet's highlightOptions.
legendOptions	Options for the legend.
...	Additional arguments passed to other functions.

Value

The modified leaflet map object with the added layer.

Examples

```
# use adcode,adcode can be obtained from leafletZH::china_city
library(leaflet)
library(leaflet.extras)
library(leafletZH)
library(sf)
data <- data.frame(adcode = seq(110101, 110110, 1), value = runif(5))
leaflet() |>
  leafletZH::addTilesAmap() |>
  addCityShape(
    data = data, adcode = "adcode", valueProperty = "value",
    popupProps = c("value")
  ) |>
  setView(lng = 116, lat = 40, zoom = 8)
```

---

addProvinceShape *Adds a choropleth map layer for provinces with additional customization options.*

Description

Adds a choropleth map layer for provinces with additional customization options.

Usage

```
addProvinceShape(
  map,
  data,
  adcode = NULL,
  provinceName = NULL,
  layerId = NULL,
  group = NULL,
  valueProperty = NULL,
  labelProperty = NULL,
  labelOptions = leaflet::labelOptions(),
  popupProps = NULL,
  popupOptions = leaflet::popupOptions(),
  scale = c("white", "red"),
  steps = 5,
  mode = "q",
  channelMode = c("rgb", "lab", "hsl", "lch"),
  padding = NULL,
  correctLightness = FALSE,
  bezierInterpolate = FALSE,
  colors = NULL,
```

```
  stroke = TRUE,
  color = "#ffffff",
  weight = 1,
  opacity = 0.5,
  fillOpacity = 0.7,
  dashArray = NULL,
  smoothFactor = 1,
  noClip = FALSE,
  pathOptions = leaflet::pathOptions(),
 highlightOptions = leaflet::highlightOptions(weight = 2, color = "#000000", fillOpacity
    = 1, opacity = 1, bringToFront = TRUE, sendToBack = TRUE),
  legendOptions = NULL,
  ...
)
```

Arguments

map	The leaflet map object to add the layer to.
data	A data frame containing the data to be visualized.
adcode	China administrative division code
provinceName	A string specifying the column name in the data frame that corresponds to the province names.
layerId	An optional string to identify the layer.
group	An optional string for grouping data.
valueProperty	The property in the geojson data that corresponds to the value to be mapped.
labelProperty	The property in the geojson data that will be used for labels.
labelOptions	Options for labels, defaults to leaflet's labelOptions.
popupProps	A named vector of properties to display in the popup.
popupOptions	Options for popups, defaults to leaflet's popupOptions.
scale	A vector of colors to use for the scale of the choropleth map.
steps	The number of steps for the color scale.
mode	The mode for the color scale, can be "q" for quantile, "e" for equal interval, etc.
channelMode	The color channel mode, can be "rgb", "lab", "hsl", or "lch".
padding	Optional padding for the choropleth layer.
correctLightness	A logical value to correct lightness for color scales.
bezierInterpolate	Whether to use bezier interpolation for the lines.
colors	An optional vector of colors to override the default color scale.
stroke	Whether to draw the stroke along the paths.
color	The color for the paths, defaults to white.
weight	The weight for the paths.

opacity	The opacity for the paths.
fillOpacity	The fill opacity for the paths.
dashArray	An optional array to create dashed lines.
smoothFactor	A factor to smooth the factor for the paths.
noClip	Whether to disable clipping of the paths.
pathOptions	Additional options for the paths, defaults to leaflet's pathOptions.
highlightOptions	Options for highlighting features, defaults to leaflet's highlightOptions.
legendOptions	Options for the legend.
...	Additional arguments passed to other functions.

Value

The modified leaflet map object with the added layer.

Examples

```
# Plot using province name, match using first two words of field
library(leaflet)
library(leaflet.extras)
library(leafletZH)
data <- data.frame(name = leafletZH::china_province$name, value = runif(34))
backg <- htmltools::tags$style(".leaflet-container { background: #000; }")
leaflet() |>
  addProvinceShape(
    data = data, provinceName = "name", valueProperty = "value",
    popupProps = c("value")
  ) |>
  setView(lng = 110, lat = 40, zoom = 4) |>
  htmlwidgets::prependContent(backg)


# Use adcode to match, adcode can be obtained in leafletZH::china_province
library(leaflet)
library(leaflet.extras)
library(leafletZH)
library(sf)
data <- data.frame(adcode = seq(110000, 150000, 10000), value = runif(5))
leaflet() |>
  leafletZH::addTilesAmap() |>
  addProvinceShape(
    data = data, adcode = "adcode", valueProperty = "value",
    popupProps = c("value")
  ) |>
  setView(lng = 110, lat = 40, zoom = 4)
```

addTilesAmap *Adds a tile layer from Amap to a leaflet map.*

Description

This function adds a tile layer from Amap to a leaflet map object.

Usage

```
addTilesAmap(
  map,
  attribution = "&copy; <a href=\"http://amap.com\">amap.com</a >",
  ...
)
```

Arguments

map	A leaflet map object to which the tile layer will be added.
attribution	A string containing the attribution text to be displayed on the map. It defaults to "© amap.com".
...	Additional arguments to be passed to the ‘leaflet::addTiles‘ function.

Value

The leaflet map object with the added tile layer.

Examples

```
library(leaflet)
leaflet() %>%
  addTilesAmap() %>%
  setView(
    lng = 120.33739,
    lat = 31.13533,
    zoom = 3
  )
```

areaCalculator *Calculate the area of a polygon defined by latitude and longitude points*

Description

This function takes in latitude and longitude vectors and calculates the area of the polygon defined by those points. It can handle different coordinate systems such as WGS-84, GCJ-02, and BD-09.

Usage

```
areaCalculator(longitude, latitude, coordinate = "WGS-84", chull = TRUE)
```

Arguments

longitude	A numeric vector of longitude points.
latitude	A numeric vector of latitude points.
coordinate	A string indicating the coordinate system of the input points, can be "WGS-84", "GCJ-02", or "BD-09". Default is "WGS-84".
chull	A logical value indicating whether to use the convex hull of the points. Default is TRUE.

Value

A numeric value representing the area of the polygon in square meters.

Examples

```
area <- areaCalculator(
  longitude = c(121.0, 122.1, 121.2, 122.15, 121.2),
  latitude =  c(31.1, 31.919, 31.917, 31.15, 31.12), coordinate = "WGS-84"
)
```

china_city *city data for China*

Description

This dataset contains spatial (sf) data for city in China, with various attributes specific to each district.

Usage

```
china_city
```

Format

An object of class `sf` (inherits from `data.frame`) with 476 rows and 10 columns.

Source

http://datav.aliyun.com/tools/atlas/

name The name of the district

adcode The administrative code for the district, a unique identifier (e.g., "110101")

childrenNum The number of lower-level administrative divisions within the district (usually 0 for districts)

level The administrative level of the area, which is "district" for all entries in this dataset

subFeatureIndex An index representing the sub-features within the district

centroid The geographical centroid of the district, represented as a string of coordinates

center The center point of the district, also represented as a string of coordinates

parent A JSON string representing the parent administrative entity, usually the province-level data

acroutes A JSON array of administrative codes that represent the full administrative hierarchy leading to the district

geometry Spatial geometry of the district, stored as an sf object in MULTIPOLYGON format

---

`china_province` *province data for China*

Description

This dataset contains spatial (sf) data for provinces in China, including various attributes related to each province.

Usage

```
china_province
```

Format

An object of class `sf` (inherits from `data.frame`) with 34 rows and 10 columns.

Source

http://datav.aliyun.com/tools/atlas/

name The name of the province

adcode The administrative code for the province, a unique identifier (e.g., "110000")

childrenNum The number of administrative divisions (e.g., counties) within the province

level The administrative level of the area, which is generally "province" for the entries in this dataset

subFeatureIndex An index representing the sub-features within the province

centroid The geographical centroid of the province, represented as a string of coordinates

center The center point of the province, also represented as a string of coordinates

parent A JSON string representing the parent administrative entity, usually the country-level data

acroutes A JSON array of administrative codes that represent the administrative hierarchy leading to the province

geometry Spatial geometry of the province, stored as an sf object in MULTIPOLYGON format

---

`convertCoordinates` *Convert Coordinates Between Different Systems*

Description

This function converts geographical coordinates between different coordinate systems, including WGS-84, GCJ-02 (Chinese National Standard), and BD-09 (Baidu Map).

Usage

```
convertCoordinates(latitude, longitude, from, to)
```

Arguments

`latitude`	Numeric value representing the latitude of the point to convert.
`longitude`	Numeric value representing the longitude of the point to convert.
`from`	A character string indicating the source coordinate system. Options include "WGS-84", "GCJ-02", and "BD-09".
`to`	A character string indicating the target coordinate system. Options include "WGS-84", "GCJ-02", and "BD-09".

Value

A numeric vector of length 2, containing the converted latitude and longitude.

Examples

```
# Convert WGS-84 coordinates to GCJ-02
convertCoordinates(39.90105, 116.42079, from = "WGS-84", to = "GCJ-02")

# Convert GCJ-02 coordinates to BD-09
convertCoordinates(39.90245, 116.42702, "GCJ-02", "WGS-84")

# Convert WGS-84 coordinates to BD-09
convertCoordinates(39.90105, 116.42079, "WGS-84", "BD-09")
```

Index