Package ‘mapview’

May 13, 2019

Type Package
Title Interactive Viewing of Spatial Data in R
Version 2.7.0
Date 2019-05-12
Maintainer Tim Appelhans <tim.appelhans@gmail.com>
Description Quickly and conveniently create interactive visualisations of spatial data with or without background maps. Attributes of displayed features are fully queryable via pop-up windows. Additional functionality includes methods to visualise true- and false-color raster images, bounding boxes, small multiples and 3D raster data cubes.
License GPL (>= 3) | file LICENSE
URL https://github.com/r-spatial/mapview
BugReports https://github.com/r-spatial/mapview/issues
Depends methods, R (>= 2.10)
Imports base64enc, brew, htmltools, htmlwidgets, lattice, leafem, leaflet (>= 2.0.0), leafpop, png, raster, Rcpp (>= 0.11.3), satellite, scales (>= 0.2.5), sf, sp, svglite, uuid, viridisLite, webshot
Suggests covr, dplyr, knitr, lwgeom, plainview, rmarkdown, stars, testthat
LinkingTo Rcpp
ByteCompile yes
LazyData TRUE
Encoding UTF-8
RoxygenNote 6.1.0
SystemRequirements GNU make
NeedsCompilation yes
Author  Tim Appelhans [cre, aut],  
       Florian Detsch [aut],  
       Christoph Reudenbach [aut],  
       Stefan Woellauer [aut],  
       Spaska Forteva [ctb],  
       Thomas Nauss [ctb],  
       Edzer Pebesma [ctb],  
       Kenton Russell [ctb],  
       Michael Sumner [ctb],  
       Jochen Darley [ctb],  
       Pierre Roudier [ctb],  
       Patrick Schratz [ctb],  
       Environmental Informatics Marburg [ctb]  

Repository  CRAN  

Date/Publication  2019-05-13 14:00:03 UTC  

R topics documented:  

   mapview-package  ............................................................ 3  
   +  .............................................................. 3  
   addFeatures  .......................................................... 4  
   addHomeButton  ......................................................... 5  
   addImageQuery  .......................................................... 6  
   addLogo  ................................................................. 7  
   addMouseCoordinates  .................................................... 7  
   addStarsImage  ........................................................... 8  
   addStaticLabels  .......................................................... 9  
   breweries  ............................................................... 11  
   coords2JSON  ............................................................ 11  
   coords2Lines  ............................................................ 12  
   coords2Polygons  ......................................................... 13  
   cubeView  ............................................................... 14  
   cubeViewOutput  .......................................................... 16  
   franconia  .............................................................. 16  
   garnishMap  ............................................................. 17  
   knit_print.mapview  ....................................................... 18  
   latticeView  ............................................................. 18  
   mapshot  ................................................................. 19  
   mapView  ................................................................. 20  
   mapview-class  .......................................................... 30  
   mapview-deprecated  ....................................................... 30  
   mapviewColors  ........................................................... 31  
   mapviewOptions  .......................................................... 32  
   mapviewOutput  ............................................................ 34  
   npts  ................................................................. 35  
   plainView  ............................................................... 35  
   poppendorf  ............................................................. 37
mapview-package

Description

Interactive viewing of spatial objects in R

Details

The package provides functionality to view spatial objects interactively. The intention is to provide interactivity for easy and quick visualization during spatial data analysis. It is not intended for fine-tuned presentation quality map production.

Author(s)

Tim Appelhans, Florian Detsch, Chris Reudenbach, Stephan Woellauer, Spaska Forteva, Thomas Nauss, Environmental Informatics Marburg

Maintainer: Tim Appelhans <tim.appelhans@gmail.com>

+ mapview + mapview adds data from the second map to the first

Description

mapview + mapview adds data from the second map to the first
mapview + data adds spatial data (raster*, sf*, sp*) to a mapview map
mapview + NULL returns the LHS map
[...]
addFeatures

**Usage**

applyFeatures method for signature 'mapview, mapview'

```r
e1 + e2
```

applyFeatures method for signature 'mapview, ANY'

```r
e1 + e2
```

applyFeatures method for signature 'mapview, `NULL`'

```r
e1 + e2
```

applyFeatures method for signature 'mapview, character'

```r
e1 + e2
```

**Arguments**

- `e1` a leaflet or mapview map to which `e2` should be added.
- `e2` a (spatial) object to be added or a mapview object from which the objects should be added to `e1`.

**Examples**

```r
m1 <- mapView(franconia, col.regions = "red")
m2 <- mapView(breweries)

### add two mapview objects
m1 + m2
'+'(m2, m1)

### add layers to a mapview object
if (interactive()) {
  library(plainview)
  m1 + breweries + plainview::poppendorf[[4]]
}
```

---

**addFeatures**

Type agnostic version of leaflet::add* functions.

**Description**

This function is deprecated. Please use leafem::addFeatures instead.

**Usage**

```r
addFeatures(map, data, pane = "overlayPane", ...)
```
**addHomeButton**

**Arguments**

- **map**  
  A leaflet or mapview map.
- **data**  
  A sf object to be added to the map.
- **pane**  
  The name of the map pane for the features to be rendered in.
- ...  
  Further arguments passed to the respective leaflet::add* functions. See addCircleMarkers, addPolylines and addPolygons.

**Description**

These functions are deprecated. Please use leafem::addHomeButton and leafem::removeHomeButton instead.

**Usage**

```
addHomeButton(map, ext, layer.name = "layer", position = "bottomright",  
              add = TRUE)
```

```
removeHomeButton(map)
```

**Arguments**

- **map**  
  a mapview or leaflet object.
- **ext**  
  the `extent`/bbox to zoom to.
- **layer.name**  
  the name of the layer to be zoomed to (or any character string)
- **position**  
  the position of the button (one of 'topleft', 'topright', 'bottomleft', 'bottomright'). Defaults to 'bottomright'.
- **add**  
  logical. Whether to add the button to the map (mainly for internal use).

**Functions**

- removeHomeButton: remove a homeButton from a map
addImageQuery

Add image query functionality to leaflet/mapview map.

Description

Add image query functionality to leaflet/mapview map.

Usage

```r
addImageQuery(map, x, band = 1, group = NULL, layerId = NULL,
              project = TRUE, type = c("mousemove", "click"), digits,
              position = "topright", prefix = "Layer", ...)
```

Arguments

- `map`: the map with the RasterLayer to be queried.
- `x`: the RasterLayer that is to be queried.
- `band`: for stars layers, the band number to be queried.
- `group`: the group of the RasterLayer to be queried.
- `layerId`: the layerId of the RasterLayer to be queried. Needs to be the same as supplied in `addRasterImage` or `link{addRastImage}`.
- `project`: whether to project the RasterLayer to conform with leaflets expected crs.Defaults to TRUE and things are likely to go haywire if set to FALSE.
- `type`: whether query should occur on 'mousemove' or 'click'. Defaults to 'mousemove'.
- `digits`: the number of digits to be shown in the display field.
- `position`: where to place the display field. Default is 'topright'.
- `prefix`: a character string to be shown as prefix for the layerId.
- ... currently not used.

Details

This function is deprecated. Please use leafem::addImageQuery instead.
addLogo

add a local or remote image (png, jpg, gif, bmp, ...) to a leaflet map

Description

This function is deprecated. Please use leafem::addLogo instead.

Usage

addLogo(map, img, alpha = 1, src = c("remote", "local"), url,
        position = c("topleft", "topright", "bottomleft", "bottomright"),
        offset.x = 50, offset.y = 13, width = 60, height = 60)

Arguments

map a mapview or leaflet object.
img the image to be added to the map.
alpha opacity of the added image.
src character specifying the source location ("local" for images from the disk, "remote" for web image sources).
url an optional URL to be opened when clicking on the image (e.g. company’s homepage).
position one of "topleft", "topright", "bottomleft", "bottomright".
offset.x the offset in x direction from the chosen position (in pixels).
offset.y the offset in y direction from the chosen position (in pixels).
width width of the rendered image in pixels.
height height of the rendered image in pixels.

addMouseCoordinates

Add mouse coordinate information at top of map.

Description

These functions are deprecated. Please use leafem::addMouseCoordinates and leafem::removeMouseCoordinates instead.

Usage

addMouseCoordinates(map, epsg = NULL, proj4string = NULL,
                     native.crs = FALSE)

removeMouseCoordinates(map)
Arguments

map
epsg
proj4string
native.crs

a mapview or leaflet object.
the epsg string to be shown.
the proj4string to be shown.
logical. whether to use the native crs in the coordinates box.

Functions

- removeMouseCoordinates: remove mouse coordinates information from a map

---

**addStarsImage**

*Add stars layer to a leaflet map*

Description

Add stars layer to a leaflet map

Usage

addStarsImage(map, x, band = 1, colors = "Spectral", opacity = 1,
               attribution = NULL, layerId = NULL, group = NULL,
               project = FALSE, method = c("bilinear", "ngb"), maxBytes = 4 * 1024
               * 1024)

Arguments

map
x
band
colors
opacity
attribution
layerId
group
project

a mapview or leaflet object.
a stars layer.
the band number to be plotted.
the color palette (see colorNumeric) or function to use to color the raster values (hint: if providing a function, set na.color to "#00000000" to make NA areas transparent)
the base opacity of the raster, expressed from 0 to 1
the HTML string to show as the attribution for this layer
the layer id
the name of the group this raster image should belong to (see the same parameter under addTiles)
if TRUE, automatically project x to the map projection expected by Leaflet (EPSG:3857); if FALSE, it's the caller's responsibility to ensure that x is already projected, and that extent(x) is expressed in WGS84 latitude/longitude coordinates
addStaticLabels

the method used for computing values of the new, projected raster image. "bilinear" (the default) is appropriate for continuous data, "ngb" - nearest neighbor - is appropriate for categorical data. Ignored if project = FALSE. See projectRaster for details.

maxBytes
the maximum number of bytes to allow for the projected image (before base64 encoding); defaults to 4MB.

Details
This is an adaption of addRasterImage. See that documentation for details.

Examples

## Not run:
library(stars)
library(leaflet)
tif = system.file("tif/L7_ETMs.tif", package = "stars")
x = read_stars(tif)
leaflet() %>%
  addProviderTiles("OpenStreetMap") %>%
  addStarsImage(x, project = TRUE)

## End(Not run)

---

addStaticLabels  Add static labels to leaflet or mapview objects

Description

Being a wrapper around addLabelOnlyMarkers, this function provides a smart-and-easy solution to add custom text labels to an existing leaflet or mapview map object.

Usage

addStaticLabels(map, data, label, group = NULL, layerId = NULL, ...)

Arguments

map  A leaflet or mapview object.
data  A sf or Spatial* object used for label placement, defaults to the locations of the first dataset in `map`.
label  The labels to be placed at the positions indicated by `data` as character, or any vector that can be coerced to this type.
group  the group of the static labels layer.
layerId  the layerId of the static labels layer.
...  Additional arguments passed to labelOptions.
Value

A labelled **mapview** object.

Author(s)

Florian Detsch

See Also

*addLabelOnlyMarkers.*

Examples

```r
## Not run:
## leaflet label display options
library(leaflet)

lopt = labelOptions(noHide = TRUE,
                     direction = 'top',
                     textOnly = TRUE)

## point labels
m1 = mapview(breweries)
l1 = addStaticLabels(m1,
                     label = breweries$number.of.types,
                     labelOptions = lopt)
l1

## polygon centroid labels
m2 = mapview(franconia)
l2 = addStaticLabels(m2,
                     label = franconia$NAME_ASCI,
                     labelOptions = lopt)
l2

## custom labels
m3 = m2 + m1
l3 = addStaticLabels(m3,
                     data = franconia,
                     label = franconia$NAME_ASCI,
                     labelOptions = lopt)
l3

## End(Not run)
```
**Selected breweries in Franconia**

**Description**

Selected breweries in Franconia

**Format**

sf feature collection POINT

**Details**

This dataset contains selected breweries in Franconia. It is partly a subset of a larger database that was compiled by students at the University of Marburg for a seminar called "The Geography of Beer: sustainability in the food industry" and partly consists of breweries downloaded from http://www.bierwandern.de/inhalt/brauereilliste.html with the kind permission of Rainer Kastl. Note that use of these data is restricted to non-commercial use and that they are explicitly excluded from the GPL license that mapview is licensed under.

**coords2JSON**

Convert a vector/matrix of coordinates to JSON format

**Description**

Similar to toJson from jsonlite, this function takes a set of coordinates as input and converts them to proper JSON format. Note that the function is powered by Rcpp which makes it a convenient alternative to existing methods when it comes to processing big datasets.

**Usage**

```r
## S4 method for signature 'numeric'
coords2JSON(x)

## S4 method for signature 'character'
coords2JSON(x, xy = c(1, 2))

## S4 method for signature 'matrix'
coords2JSON(x, xy = c(1, 2))
```

**Arguments**

- `x` A 'numeric' vector with a single pair of coordinates or a matrix with multiple pairs of input coordinates, typically projected in EPSG:4326 (http://spatialreference.org/ref/epsg/wgs-84/).
- `xy` An 'integer' vector specifying the coordinate columns.
Value

A single 'character' object in JSON format.

Author(s)

Florian Detsch

Examples

crd <- matrix(ncol = 3, nrow = 12)

# x-coordinates
set.seed(10)
crd[, 1] <- rnorm(nrow(crd), 10, 3)

# y-coordinates
set.seed(10)
crd[, 2] <- rnorm(nrow(crd), 50, 3)

# additional data
crd[, 3] <- month.abb

# reformat a single pair of coordinates
coords2JSON(crd[1, ])

# reformat multiple pairs of coordinates at once
coords2JSON(crd)

---

coords2Lines Convert points to SpatialLines *

Description

Create a SpatialLines* object from a Line object or set of point coordinates in one go, i.e. without being required to run through the single steps outlined in SpatialLines.

Usage

## S4 method for signature 'matrix'
coords2Lines(coords, ID, data, match.ID = TRUE, ...)

## S4 method for signature 'Line'
coords2Lines(coords, ID, data, match.ID = TRUE, ...)
coords2Polygons

Arguments

- **coords**: Line object or 2-column numeric matrix with x and y coordinates.
- **ID**: character, see `Lines`.
- **data**: data.frame with data to add to the output SpatialLines* object (optional).
- **match.ID**: logical, see `SpatialLinesDataFrame`.

Value

If data is missing, a SpatialLines object; else a SpatialLinesDataFrame object.

See Also

`SpatialLines-class`, `SpatialLinesDataFrame`.

Examples

```r
library(sp)

coords1 <- cbind(c(2, 4, 4, 1, 2), c(2, 3, 5, 4, 2))
sln1 <- coords2Lines(coords1, ID = "A")

coords2 <- cbind(c(5, 4, 2, 5), c(2, 3, 2, 2))
sln2 <- coords2Lines(coords2, ID = "B")

mapview(sln1)

plot(sln1, col = "grey75")
plot(sln2, col = "grey25", add = TRUE)
```

coords2Polygons  Convert points to SpatialPolygons*

Description

Create a SpatialPolygons* object from a Polygon object or set of point coordinates in one go, i.e. without being required to run through the single steps outlined in `SpatialPolygons`.

Usage

```r
## S4 method for signature 'matrix'
coords2Polygons(coords, hole = NA, ID, data,
                 match.ID = TRUE, ...)

## S4 method for signature 'Polygon'
coords2Polygons(coords, ID, data, match.ID = TRUE,
                 ...)
Arguments

- **coords**  Polygon object or 2-column numeric matrix with x and y coordinates.
- **hole**  logical, see *Polygon*.
- **ID**  character, see *Polygons*.
- **data**  data.frame with data to add to the output SpatialPolygons* object (optional).
- **match.ID**  logical, see *SpatialPolygonsDataFrame*.
- ...  Further arguments passed on to *SpatialPolygons* (i.e., p0 and proj4string).

Value

If data is missing, a SpatialPolygons object; else a SpatialPolygonsDataFrame object.

See Also

*SpatialPolygons-class, SpatialPolygonsDataFrame*.

Examples

```r
library(sp)

coords1 <- cbind(c(2, 4, 4, 1, 2), c(2, 3, 5, 4, 2))
spy1 <- coords2Polygons(coords1, ID = "A")

coords2 <- cbind(c(5, 4, 2, 5), c(2, 3, 2, 2))
spy2 <- coords2Polygons(coords2, ID = "B")

plot(spy1, col = "grey75")
plot(spy2, col = "grey25", add = TRUE)
```

---

**cubeView**

*View a RasterStack or RasterBrick as 3-dimensional data cube.*

Description

Create a 3D data cube from a RasterStack or RasterBrick. The cube can be freely rotated so that Hovmoller views of x - z and y - z are possible.

Usage

```r
cubeView(x, at, col.regions = mapviewGetOption("raster.palette"),
          na.color = mapviewGetOption("na.color"), legend = TRUE)
cubeview(...)
```
cubeView

Arguments

x  a RasterStack or RasterBrick
at  the breakpoints used for the visualisation. See levelplot for details.
col.regions  color (palette). See levelplot for details.
na.color  color for missing values.
legend  logical. Whether to plot a legend.
...  currently not used.

Details

The visible layers are alterable by keys:
x-axis: LEFT / RIGHT arrow key
y-axis: DOWN / UP arrow key
z-axis: PAGE_DOWN / PAGE_UP key

Note: In RStudio cubeView may show a blank viewer window. In this case open the view in a web-browser (RStudio button at viewer: "show in new window").
Note: Because of key focus issues key-press-events are not always recognised within RStudio at Windows. In this case open the view in a web-browser (RStudio button at viewer: "show in new window").
Press and hold left mouse-button to rotate the cube. Press and hold right mouse-button to move the cube. Spin mouse-wheel or press and hold middle mouse-button and move mouse down/up to zoom the cube.

Functions

• cubeview: alias for ease of typing

Author(s)

Stephan Woellauer and Tim Appelhans

Examples

## Not run:
library(raster)

kili_data <- system.file("extdata", "kiliNDVI.tif", package = "mapview")
kiliNDVI <- stack(kili_data)

cubeView(kiliNDVI)

clr <- viridisLite::viridis
cubeView(kiliNDVI, at = seq(-0.15, 0.95, 0.1), col.regions = clr)

## End(Not run)
cubeViewOutput  

**Widget output function for use in Shiny**

**Description**

Widget output function for use in Shiny

**Usage**

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

**Arguments**

- `outputId`: Output variable to read from
- `width`, `height`: The width and height of the map (see `shinyWidgetOutput`)

**Description**

Administrative district borders of Franconia

**Format**

`sf` feature collection MULTIPOLYGON

**Details**


**Source**

**garnishMap**

Garnish/decorate leaflet or mapview maps.

---

**Description**

This function provides a versatile interface to add components to a leaflet or mapview map. It takes functions such as "addMouseCoordinates" or `addLayersControl` and their respective arguments and adds them to the map. Arguments must be named. Functions can be plain or character strings.

**Usage**

```r
garnishMap(map, ...)```

**Arguments**

- `map`: a mapview or leaflet object.
- `...`: functions and their arguments to add things to a map.

**Examples**

```r
text <- 
```

```
library(leaflet)
library(leafem)
library(leafpop)
```

```
m <- leaflet() 
%>%
addProviderTiles("OpenStreetMap")
garnishMap(m, leafem::addMouseCoordinates, style = "basic")
```

```r
## add more than one with named argument
library(raster)
```

```
m1 <- garnishMap(m, leafem::addMouseCoordinates, leafem::addHomeButton, 
                  ext = extent(breweries))
m1
```

```r
## even more flexible
m2 <- garnishMap(m1, addPolygons, data = franconia, 
                  popup = leafpop::popupTable(franconia), 
                  fillOpacity = 0.8, color = "black", fillColor = "#BEBEBE")
garnishMap(m2, addCircleMarkers, data = breweries)
```

```r
text <- 
```
knit_print.mapview  Print functions for mapview objects used in knitr

Description
Print functions for mapview objects used in knitr

Usage
knit_print.mapview(x, ...)

Arguments
x  A mapview object
... further arguments passed on to knit_print

latticeView  View two or more (possibly synchronised) mapview or leaflet maps

Description
These functions are deprecated. Please use leafsync::sync and leafsync::latticeView instead.

Usage
latticeView(..., ncol = 2, sync = "none", sync.cursor = FALSE,
        no.initial.sync = TRUE)
latticeView(...)
sync(..., ncol = 2, sync = "all", sync.cursor = TRUE,
        no.initial.sync = TRUE)

Arguments
... any number of mapview or leaflet objects or a list thereof
ncol how many columns should be plotted
sync whether to synchronise zoom and pan for certain elements. Possible values are "all" (default) to sync all maps, "none" to disable synchronisation or a list of panel numbers, e.g. list(c(1, 3), c(2, 4)) will synchronise panels 1 & 3 and panels 2 & 4. Panels are drawn from top right to bottom left.
sync.cursor whether to show cursor position in synced panels (default TRUE).
no.initial.sync whether to sync the initial view (default TRUE).
mapshot

Functions

- `latticeview`: alias for ease of typing
- `sync`: convenience function for syncing maps

---

mapshot  

*Save mapview or leaflet map as HTML and/or image*

---

Description

Save a mapview or leaflet map as `.html` index file or `.png`, `.pdf`, or `.jpeg` image.

Usage

```r
mapshot(x, url = NULL, file = NULL, remove_url = TRUE,
         remove_controls = c("zoomControl", "layersControl", "homeButton",
         "scaleBar"), ...)
```

Arguments

- `x`: mapview or leaflet object.
- `url`: Output `.html` file. If not supplied and `file` is specified, a temporary index file will be created.
- `file`: Output `.png`, `.pdf`, or `.jpeg` file.
- `remove_url`: logical. If `TRUE` (default), the `.html` file is removed once processing is completed. Only applies if `url` is not specified.
- `remove_controls`: character vector of control buttons to be removed from the map when saving to file. Any combination of "zoomControl", "layersControl", "homeButton", "scaleBar". If set to `NULL` nothing will be removed.
- `...`: Further arguments passed on to `webshot`.

Details

mapshot can be used to save both leaflet and mapview maps as html or png files or both.

NOTE 1: In case you want to save larger maps produced with mapview (i.e. if you see the following warning: "the supplied feature layer has more points/vertices than the set threshold. using special rendering function, hence things may not behave as expected from a standard leaflet map") mapshot is likely to fail. Try setting `selfcontained = FALSE` to avoid errors and create a valid local html file.

NOTE 2: In case you want to save a map with popupGraphs or popupImages the respective graph/image files will be located one level above the specified target location. In case you want to move the html file, make sure to also move the respective *.graphs* folder one level above.
mapView

View spatial objects interactively

Description

this function produces an interactive view of the specified spatial object(s) on top of the specified base maps.

Usage

```r
## S4 method for signature 'RasterLayer'
mapView(x, map = NULL,
maxpixels = mapViewGetOption("mapview.maxpixels"),
col.regions = mapViewGetOption("raster.palette")(256), at = NULL,
na.color = mapViewGetOption("na.color"), use.layer.names = FALSE,
values = NULL, map.types = mapViewGetOption("basemaps"),
alpha.regions = 0.8, legend = mapViewGetOption("legend"),
legend.opacity = 1, trim = TRUE,
verbose = mapViewGetOption("verbose"), layer.name = NULL,
homebutton = TRUE, native.crs = FALSE, method = c("bilinear",
"ngb"), label = TRUE, query.type = c("mousemove", "click"),
query.digits, query.position = "topright", query.prefix = "Layer",
viewer.suppress = FALSE, ...)
```
## S4 method for signature 'stars'
mapView(x, band = 1, map = NULL,
  maxpixels = mapViewGetOption("mapview.maxpixels"),
  col.regions = mapViewGetOption("raster.palette") (256), at = NULL,
  na.color = mapViewGetOption("na.color"), use.layer.names = FALSE,
  values = NULL, map.types = mapViewGetOption("basemaps"),
  alpha.regions = 0.8, legend = mapViewGetOption("legend"),
  legend.opacity = 1, trim = TRUE,
  verbose = mapViewGetOption("verbose"), layer.name = NULL,
  homebutton = TRUE, native.crs = FALSE, method = c("bilinear",
  "ngb"), label = TRUE, query.type = c("mousemove", "click"),
  query.digits, query.position = "topright", query.prefix = "Layer",
  viewer.suppress = FALSE, ...)

## S4 method for signature 'RasterStackBrick'
mapView(x, map = NULL,
  maxpixels = mapViewGetOption("mapview.maxpixels"),
  col.regions = mapViewGetOption("raster.palette") (256), at = NULL,
  na.color = mapViewGetOption("na.color"), use.layer.names = TRUE,
  values = NULL, map.types = mapViewGetOption("basemaps"),
  legend = mapViewGetOption("legend"), legend.opacity = 1,
  trim = TRUE, verbose = mapViewGetOption("verbose"),
  homebutton = TRUE, method = c("bilinear", "ngb"), label = TRUE,
  query.type = c("mousemove", "click"), query.digits,
  query.position = "topright", query.prefix = "Layer",
  viewer.suppress = FALSE, ...)

## S4 method for signature 'Satellite'
mapView(x, map = NULL,
  maxpixels = mapViewGetOption("mapview.maxpixels"),
  col.regions = mapViewGetOption("raster.palette") (256), at = NULL,
  na.color = mapViewGetOption("na.color"), values = NULL,
  map.types = mapViewGetOption("basemaps"),
  legend = mapViewGetOption("legend"), legend.opacity = 1,
  trim = TRUE, verbose = mapViewGetOption("verbose"),
  homebutton = TRUE, method = c("bilinear", "ngb"), label = TRUE,
  ...)

## S4 method for signature 'sf'
mapView(x, map = NULL, pane = "auto",
  canvas = useCanvas(x), viewer.suppress = canvas, zcol = NULL,
  burst = FALSE, color = mapViewGetOption("vector.palette"),
  col.regions = mapViewGetOption("vector.palette"), at = NULL,
  na.color = mapViewGetOption("na.color"), cex = 6,
  lwd = lineWidth(x), alpha = 0.9, alpha.regions = regionOpacity(x),
  na.alpha = regionOpacity(x), map.types = NULL,
  verbose = mapViewGetOption("verbose"),
  popup = leafpop::popupTable(x), layer.name = NULL,
label = makeLabels(x, zcol), legend = mapViewGetOption("legend"),
legend.opacity = 1, homebutton = TRUE, native.crs = FALSE,
highlight = mapViewHighlightOptions(x, alpha.regions, alpha, lwd),
maxpoints = getMaxFeatures(x), ...)

## S4 method for signature 'sfc'
mapView(x, map = NULL, pane = "auto",
canvas = useCanvas(x), viewer.suppress = canvas,
color = standardColor(x), col.regions = standardColRegions(x),
at = NULL, na.color = mapViewGetOption("na.color"), cex = 6,
lwd = lineWidth(x), alpha = 0.9, alpha.regions = regionOpacity(x),
map.types = NULL, verbose = mapViewGetOption("verbose"),
popup = NULL, layer.name = deparse(substitute(x, env =
parent.frame())), label = makeLabels(x),
legend = mapViewGetOption("legend"), legend.opacity = 1,
homebutton = TRUE, native.crs = FALSE,
highlight = mapViewHighlightOptions(x, alpha.regions, alpha, lwd),
maxpoints = getMaxFeatures(x), ...)

## S4 method for signature 'numeric'
mapView(x, y, type = "p", grid = TRUE, label, ...)

## S4 method for signature 'data.frame'
mapView(x, xcol, ycol, grid = TRUE, aspect = 1,
popup = leafpop::popupTable(x), label, crs = NA, ...)

## S4 method for signature 'XY'
mapView(x, map = NULL, pane = "auto",
canvas = useCanvas(x), viewer.suppress = canvas,
color = standardColor(x), col.regions = standardColRegions(x),
at = NULL, na.color = mapViewGetOption("na.color"), cex = 6,
lwd = lineWidth(x), alpha = 0.9, alpha.regions = regionOpacity(x),
map.types = NULL, verbose = mapViewGetOption("verbose"),
popup = NULL, layer.name = deparse(substitute(x, env =
parent.frame())), label = makeLabels(x),
legend = mapViewGetOption("legend"), legend.opacity = 1,
homebutton = TRUE, native.crs = FALSE,
highlight = mapViewHighlightOptions(x, alpha.regions, alpha, lwd),
maxpoints = getMaxFeatures(x), ...)

## S4 method for signature 'XYZ'
mapView(x, layer.name = deparse(substitute(x, env =
parent.frame())), ...)

## S4 method for signature 'XYM'
mapView(x, layer.name = deparse(substitute(x, env =
parent.frame())), ...)
```r
## S4 method for signature 'XYZM'
mapView(x, layer.name = deparse(substitute(x, env = parent.frame(1))), ...)

## S4 method for signature 'bbox'
mapView(x, layer.name = deparse(substitute(x, env = parent.frame(1))), alpha.regions = 0.2, ...)

## S4 method for signature 'missing'
mapView(map.types = mapviewGetOption("basemaps"), ...)

## S4 method for signature 'list'
mapView(x, map = NULL, zcol = NULL, burst = FALSE, color = mapviewGetOption("vector.palette"), col.regions = mapviewGetOption("vector.palette"), at = NULL, na.color = mapviewGetOption("na.color"), cex = 6, lwd = lapply(x, lineWidth), alpha = lapply(seq(x), function(i) 0.9), alpha.regions = lapply(seq(x), function(i) 0.6), map.types = mapviewGetOption("basemaps"), verbose = mapviewGetOption("verbose"), popup = lapply(seq(x), function(i) { leafpop::popupTable(x[[i]]) }), layer.name = deparse(substitute(x, env = parent.frame())), label = lapply(seq(x), function(i) { makeLabels(x[[i]], zcol = zcol[[i]]) }), legend = mapviewGetOption("legend"), legend.opacity = 1, homebutton = TRUE, native.crs = FALSE, maxpoints = NULL, ...)

## S4 method for signature 'ANY'
mapview(...)  

## S4 method for signature 'SpatialPixelsDataFrame'
mapView(x, map = NULL, zcol = NULL, maxpixels = mapviewGetOption("mapview.maxpixels"), col.regions = mapviewGetOption("raster.palette") (256), at = NULL, na.color = mapviewGetOption("na.color"), use.layer.names = FALSE, values = NULL, map.types = mapviewGetOption("basemaps"), alpha.regions = 0.8, legend = mapviewGetOption("legend"), legend.opacity = 1, trim = TRUE, verbose = mapviewGetOption("verbose"), layer.name = NULL, homebutton = TRUE, native.crs = FALSE, method = c("bilinear", "ngb"), label = TRUE, query.type = c("mousemove", "click"), query.digits, query.position = "topright", query.prefix = "Layer", viewer.suppress = FALSE, ...)

## S4 method for signature 'SpatialGridDataFrame'
mapView(x, map = NULL, zcol = NULL, maxpixels = mapviewGetOption("mapview.maxpixels"),
```
col.regions = mapviewGetOption("rasterpalette")(256), at = NULL, 
na.color = mapviewGetOption("na.color"), use.layer.names = FALSE, 
values = NULL, map.types = mapviewGetOption("basemaps"), 
alpha.regions = 0.8, legend = mapviewGetOption("legend"), 
legend.opacity = 1, trim = TRUE, 
verbose = mapviewGetOption("verbose"), layer.name = NULL, 
homebutton = TRUE, native.crs = FALSE, method = c("bilinear", 
"ngb"), label = TRUE, query.type = c("mousemove", "click"), 
query.digits, query.position = "topright", query.prefix = "Layer", 
viewer.suppress = FALSE, ...)

## S4 method for signature 'SpatialPointsDataFrame'
mapView(x, zcol = NULL, 
    layer.name = NULL, ...)

## S4 method for signature 'SpatialPoints'
mapView(x, zcol = NULL, layer.name = NULL, 
    ...)

## S4 method for signature 'SpatialPolygonsDataFrame'
mapView(x, zcol = NULL, 
    layer.name = NULL, ...)

## S4 method for signature 'SpatialPolygons'
mapView(x, zcol = NULL, layer.name = NULL, 
    ...)

## S4 method for signature 'SpatialLinesDataFrame'
mapView(x, zcol = NULL, 
    layer.name = NULL, ...)

## S4 method for signature 'SpatialLines'
mapView(x, zcol = NULL, layer.name = NULL, 
    ...)

Arguments

x | a Raster* or Spatial* or Satellite or sf object or a list of any combination of those. Furthermore, this can also be a data.frame, a numeric vector. If missing, a blank map will be drawn.

map | an optional existing map to be updated/added to.

maxpixels | integer > 0. Maximum number of cells to use for the plot. If maxpixels < ncell(x), sampleRegular is used before plotting.

col.regions | color (palette) pixels. See levelplot for details.
at | the breakpoints used for the visualisation. See levelplot for details.

na.color | color for missing values

use.layer.names | should layer names of the Raster* object be used?
values a vector of values for the visualisation of the layers. Per default these are calculated based on the supplied raster* object.

map.types character specifications for the base maps. see http://leaflet-extras.github.io/leaflet-providers/preview/ for available options.

alpha.regions opacity of the fills of points, polygons or raster layer(s)

legend should a legend be plotted

legend.opacity opacity of the legend

trim should the raster be trimmed in case there are NAs on the edges

verbose should some details be printed during the process

layer.name the name of the layer to be shown on the map

homebutton logical, whether to add a zoom-to-layer button to the map. Defaults to TRUE

native.crs logical whether to reproject to web map coordinate reference system (web mercator - epsg:3857) or render using native CRS of the supplied data (can also be NA). Default is FALSE which will render in web mercator. If set to TRUE now background maps will be drawn (but rendering may be much quicker as no reprojecting is necessary). Currently only works for simple features.

method for raster data only (raster/stars). Method used to compute values for the resampled layer that is passed on to leaflet. mapview does projection on-the-fly to ensure correct display and therefore needs to know how to do this projection. The default is 'bilinear' (bilinear interpolation), which is appropriate for continuous variables. The other option, 'ngb' (nearest neighbor), is useful for categorical variables. Ignored if the raster layer is of class factor in which case "ngb" is used.

label For vector data (sf/sp) a character vector of labels to be shown on mouseover. See addControl for details. For raster data (Raster*/stars) a logical indicating whether to add image query.

query.type for raster methods only. Whether to show raster value query on 'mousemove' or 'click'. Ignored if label = FALSE.

query.digits for raster methods only. The amount of digits to be shown by raster value query. Ignored if label = FALSE.

query.position for raster methods only. The position of the raster value query info box. See position argument of addLegend for possible values. Ignored if label = FALSE.

query.prefix for raster methods only. a character string to be shown as prefix for the layerId. Ignored if label = FALSE.

viewer.suppress whether to render the map in the browser (TRUE) or the RStudio viewer (FALSE). When not using RStudio, maps will open in the browser by default. This is passed to sizingPolicy via leafletSizingPolicy. For raster data the default is FALSE. For vector data it depends on argument canvas.

... additional arguments passed on to respective functions. See addRasterImage, addCircles, addPolygons, addPolylines for details

band for stars layers, the band number to be plotted.
name of the map pane in which to render features. See addMapPane for details. Currently only supported for vector layers. Ignored if canvas = TRUE. The default "auto" will create different panes for points, lines and polygons such that points overlay lines overlay polygons. Set to NULL to get default leaflet behaviour where all features are rendered in the same pane and layer order is determined by automatically/sequentially.

whether to use canvas rendering rather than svg. May help performance with larger data. See http://leafletjs.com/reference-1.3.0.html#canvas for more information. Only applicable for vector data. The default setting will decide automatically, based on feature complexity.

attribute name(s) or column number(s) in attribute table of the column(s) to be rendered. See also Details.

whether to show all (TRUE) or only one (FALSE) layer(s). See also Details.

color (palette) for points/polygons/lines

attribute name(s) or column number(s) in attribute table of the column(s) to be used for defining the size of circles

line width

opacity of lines

opacity of missing values

a list of HTML strings with the popup contents, usually created from popupTable. See addControl for details.

either FALSE, NULL or a list of styling options for feature highlighting on mouse hover. See highlightOptions for details.

the maximum number of points making up the geometry. In case of lines and polygons this refers to the number of vertices. See Details for more information.

numeric vector.

whether to render the numeric vector x as a point "p" or line "l" plot.

whether to plot a (scatter plot) xy-grid to aid interpretation of the visualisation. Only relevant for the data.frame method.

the column to be mapped to the x-axis. Only relevant for the data.frame method.

the column to be mapped to the y-axis. Only relevant for the data.frame method.

the ratio of x/y axis coordinates to adjust the plotting space to fit the screen. Only relevant for the data.frame method.

an optional crs specification for the provided data to enable rendering on a basemap. See argument description in st_sf for details.

Details

If zcol is not NULL but a length one character vector (referring to a column name of the attribute table) and burst is TRUE, one layer for each unique value of zcol will be drawn. The same will happen if burst is a length one character vector (again referring to a column of the attribute table).

NOTE: if XYZ or XYM or XYZM data from package sf is passed to mapview, dimensions Z
and M will be stripped to ensure smooth rendering even though the popup will potentially still say something like "POLYGON Z".

maxpoints is taken to determine when to switch rendering from svg to canvas overlay for performance. The threshold calculation is done as follows:
if the number of points (in case of point data) or vertices (in case of polygon or line data) > maxpoints then render using special render function. Within this render function we approximate the complexity of features by

\[
maxFeatures \leftarrow \frac{maxfeatures}{npts(data) / length(data)}
\]

where npts determines the number of points/vertices and length the number of features (points, lines or polygons). When the number of features in the current view window is larger than maxFeatures then features are rendered on the canvas, otherwise they are rendered as svg objects and fully queryable.

Methods (by class)

- stars: \texttt{stars}
- RasterStackBrick: \texttt{stack / brick}
- Satellite: \texttt{satellite}
- sf: \texttt{st_sf}
- sfc: \texttt{st_sfc}
- numeric: \texttt{numeric}
- data.frame: \texttt{data.frame}
- XY: \texttt{st_sfc}
- XYZ: \texttt{st_sfc}
- XYM: \texttt{st_sfc}
- XYZM: \texttt{st_sfc}
- bbox: \texttt{st_bbox}
- missing: initiate a map without an object
- list: \texttt{list}
- ANY: alias for ease of typing
- SpatialPixelsDataFrame: \texttt{SpatialPixelsDataFrame}
- SpatialGridDataFrame: \texttt{SpatialGridDataFrame}
- SpatialPointsDataFrame: \texttt{SpatialPointsDataFrame}
- SpatialPoints: \texttt{SpatialPoints}
- SpatialPolygonsDataFrame: \texttt{SpatialPolygonsDataFrame}
- SpatialPolygons: \texttt{SpatialPolygons}
- SpatialLinesDataFrame: \texttt{SpatialLinesDataFrame}
- SpatialLines: \texttt{SpatialLines}
Author(s)

Tim Appelhans

Examples

```r
## Not run:
mapview()

## simple features
library(sf)

# sf
mapview(breweries)
mapview(franconia)

# sfc
mapview(st_geometry(breweries)) # no popup

# sfg / XY - taken from ?sf::sf_point
outer = matrix(c(0,0,10,0,10,0,10,0,0), ncol=2, byrow=TRUE)
hole1 = matrix(c(1,1,2,2,2,1,1), ncol=2, byrow=TRUE)
hole2 = matrix(c(5,5,6,6,6,5,5), ncol=2, byrow=TRUE)
pts = list(outer, hole1, hole2)
(pl1 = st_polygon(pts))
mapview(pl1)

## raster
if (interactive()) {
  library(plainview)
  mapview(plainview::poppendorf[[5]])
}

## spatial objects
mapview(leaflet::gadmCHE)
mapview(leaflet::atlStorms2005)

## styling options & legends
mapview(franconia, color = "white", col.regions = "red")
mapview(franconia, color = "magenta", col.regions = "white")
mapview(breweries, zcol = "founded")
mapview(breweries, zcol = "founded", at = seq(1400, 2200, 200), legend = TRUE)
mapview(franconia, zcol = "district", legend = TRUE)
clrs <- sf.colors
mapview(franconia, zcol = "district", col.regions = clrs, legend = TRUE)

## multiple layers
mapview(franconia) + breweries
mapview(list(breweries, franconia))
```
mapView

mapview(franconia) + mapview(breweries) + trails

mapview(franconia, zcol = "district") + mapview(breweries, zcol = "village")
mapview(list(franconia, breweries),
    zcol = list("district", NULL),
    legend = list(TRUE, FALSE))

### burst

mapview(franconia, burst = TRUE)
mapview(franconia, burst = TRUE, hide = TRUE)
mapview(franconia, zcol = "district", burst = TRUE)

### ceci constitue la fin du pipe

library(dplyr)
library(sf)

franconia %>%
  sf::st_union() %>%
  mapview()

franconia %>%
  group_by(district) %>%
  summarize() %>%
  mapview(zcol = "district")

franconia %>%
  group_by(district) %>%
  summarize() %>%
  mutate(area = st_area(.) / 1e6) %>%
  mapview(zcol = "area")

franconia %>%
  mutate(area = sf::st_area(.)) %>%
  mapview(zcol = "area", legend = TRUE)

breweries %>%
  st_intersection(franconia) %>%
  mapview(zcol = "district")

franconia %>%
  mutate(count = lengths(st_contains(., breweries))) %>%
  mapview(zcol = "count")

franconia %>%
  mutate(count = lengths(st_contains(., breweries)),
    density = count / st_area(.) %>%
  mapview(zcol = "density")

## End(Not run)
### Class mapview

**Description**

Class mapview

**Slots**

- `object` the spatial object
- `map` the leaflet map object

### Deprecated functions in mapview

**Description**

These functions still work but will be removed (defunct) in the next version. See below for information on which package they have been moved to.

**Details**

- `cubeview`: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'cubeview'.
- `cubeView`: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'cubeview'.
- `cubeViewOutput`: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'cubeview'.
- `renderCubeView`: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'cubeview'.
- `slideview`: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'slideview'.
- `slideView`: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'slideview'.
- `slideViewOutput`: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'slideview'.
- `renderslideview`: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'slideview'.
- `latticeView`: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafsync'.
- `sync`: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafsync'.
• plainview: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'plainview'.
• plainView: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'plainview'.
• popupTable: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafpop'.
• popupImage: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafpop'.
• popupGraph: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafpop'.
• addFeatures: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafem'.
• garnishMap: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafem'.
• addHomeButton: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafem'.
• removeHomeButton: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafem'.
• addImageQuery: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafem'.
• addLogo: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafem'.
• addMouseCoordinates: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafem'.
• removeMouseCoordinates: This function is deprecated, and will be removed in the next version of this package. This function has been migrated to package 'leafem'.

mapviewColors

mapview version of leaflet::color* functions

Description

mapview version of leaflet::color* functions
Color palettes for mapview

Usage

mapviewColors(x, zcol = NULL,
    colors = mapviewGetOption("vector.palette"), at = NULL,
    na.color = mapviewGetOption("na.color"), ...)

mapviewPalette(name = "mapviewVectorColors")

mapViewPalette(name)
Arguments

x Spatial* or Raster* object
zcol the column to be colored
colors color vector to be used for coloring the levels specified in at
at numeric vector giving the breakpoints for the colors
na.color the color for NA values.
... additional arguments passed on to level.colors
name Name of the color palette to be used. One of "mapviewVectorColors" (default), "mapviewRasterColors", "mapviewSpectralColors" or "mapviewTopoColors".

Author(s)

Tim Appelhans

See Also

level.colors
colorRampPalette

Description

To permanently set any of these options, you can add them to <your R installation>/etc/Rprofile.site>. For example, to change the default number of pixels to be visualised for Raster* objects, add a line like this: options(mapviewMaxPixels = 700000) to that file.

Usage

mapviewOptions(platform, basemaps, raster.size, mapview.maxpixels, plainview.maxpixels, maxpoints, maxpolygons, maxlines, raster.palette, vector.palette, verbose, na.color, legend, legend.pos, layers.control.pos, default = FALSE, console = TRUE, leafletWidth, leafletHeight)

mapviewGetOption(param)

Arguments

platform character. The platform to be used. Current options are "leaflet" and "quickmapr".
basemaps character. The basemaps to be used for rendering data. See http://leaflet-extras.
github.io/leaflet-providers/preview/ for possible values
raster.size numeric. see the maxBytes argument in addRasterImage
mapviewOptions

mapview.maxpixels
numeric. The maximum amount of pixels allowed for Raster* objects to be rendered with mapview. Defaults to 500000. Set this higher if you have a potent machine or are patient enough to wait a little.

plainview.maxpixels
numeric. The maximum amount of pixels allowed for Raster* objects to be rendered with plainview. Defaults to 1000000. Set this higher if you have a potent machine or are patient enough to wait a little.

maxpoints numeric. Maximum number of points allowed for leaflet overlay rendering. If this number is exceeded rendering will be done using special functionality which will provide much more speed and better handling. This means that standard functionality is reduced. For example adding layers via "+" is not possible anymore.

maxpolygons numeric. Maximum number of polygons allowed for leaflet overlay rendering. If this number is exceeded rendering will be done using special functionality which will provide much more speed and better handling. This means that standard functionality is reduced. For example adding layers via "+" is not possible anymore.

maxlines numeric. Maximum number of lines allowed for leaflet overlay rendering. If this number is exceeded rendering will be done using special functionality which will provide much more speed and better handling. This means that standard functionality is reduced. For example adding layers via "+" is not possible anymore.

raster.palette a color palette function for raster visualisation. Should be a function that takes an integer as input and returns a vector of colors. See colorRampPalette for details.

vector.palette a color palette function for vector visualisation. Should be a function that takes an integer as input and returns a vector of colors. See colorRampPalette for details.

verbose logical. Many functions in mapview provide details about their behaviour. Set this to TRUE if you want to see these printed to the console.

na.color character. The default color to be used for NA values.

legend logical. Whether or not to show a legend for the layer(s).

legend.pos Where should the legend be placed? One of "topleft", "topright", "bottomleft", "bottomright".

layers.control.pos character. Where should the layer control be placed? One of "topleft", "topright", "bottomleft", "bottomright".

default logical. If TRUE all options are set to their default values

console logical. Should the options be printed to the console

leafletWidth, leafletHeight height and width of the htmlwidget in px.

param character. parameter to be queried.
Value

list of the current options (invisibly). If no arguments are provided the options are printed.

Functions

• `mapviewGetOption`: query single `mapviewOption` parameters

Author(s)

Tim Appelhans

See Also

`rasterOptions.options`

Examples

```r
mapviewOptions()
mapviewOptions(na.color = "pink")
mapviewOptions()

mapviewGetOption("platform")

mapviewOptions(default = TRUE)
mapviewOptions()
```

---

`mapviewOutput` *Create a mapview UI element for use with shiny*

Description

Create a mapview UI element for use with shiny

Usage

```r
mapviewOutput(outputId, width = "100\%", height = 400)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>outputId</code></td>
<td>Output variable to read from</td>
</tr>
<tr>
<td><code>width</code>, <code>height</code></td>
<td>the width and height of the map (see <code>shinyWidgetOutput</code>)</td>
</tr>
</tbody>
</table>
**npts**

*count the number of points/vertices/nodes of sf objects*

**Description**

count the number of points/vertices/nodes of sf objects

**Usage**

\[
npts(x, \text{by\_feature} = \text{FALSE})
\]

**Arguments**

- **x**
  - an sf/sfc object
- **by\_feature**
  - count total number of vertices (FALSE) of for each feature (TRUE).

**Note**

currently only works for *POINTS, *LINES and *POLYGONS (not GEOMETRYCOLLECTION).

**Examples**

\[
npts(franconia)
npts(franconia, \text{by\_feature} = \text{TRUE})
npts(sf::st\_geometry(franconia[1, ])) # first polygon
npts(breweries) # is the same as
nrow(breweries)
\]

---

**plainView**

*View raster objects interactively without background map but in any CRS*

**Description**

This function is deprecated. Please use plainview::plainView instead.

**Usage**

```r
## S4 method for signature 'RasterLayer'
plainView(x,
           maxpixels = mapviewGetOption("plainview.maxpixels"),
           col.regions = mapviewGetOption("raster.palette")((56), at,
           na.color = mapviewGetOption("na.color"), legend = TRUE,
           verbose = mapviewGetOption("verbose"),
```

layer.name = deparse(substitute(x, env = parent.frame())),
gdal = TRUE, ...)

## S4 method for signature 'RasterStackBrick'
plainView(x, r = 3, g = 2, b = 1,
  na.color = mapviewGetOption("na.color"),
  maxpixels = mapviewGetOption("plainview.maxpixels"),
  layer.name = deparse(substitute(x, env = parent.frame())), ...)

## S4 method for signature 'SpatialPixelsDataFrame'
plainView(x, zcol = 1, ...)

## S4 method for signature 'ANY'
plainview(...)

### Arguments

- **x**: a raster* object
- **maxpixels**: integer > 0. Maximum number of cells to use for the plot. If maxpixels < ncell(x), sampleRegular is used before plotting.
- **col.regions**: color (palette). See levelplot for details.
- **at**: the breakpoints used for the visualisation. See levelplot for details.
- **na.color**: color for missing values.
- **legend**: either logical or a list specifying any of the components described in the colorkey section of levelplot.
- **verbose**: should some details be printed during the process
- **layer.name**: the name of the layer to be shown on the map
- **gdal**: logical. If TRUE (default) gdal_translate is used to create the png file for display when possible. See details for further information.
- **...**: additional arguments passed on to respective functions. See addRasterImage, addCircles, addPolygons, addPolylines for details
- **r**: integer. Index of the Red channel, between 1 and nlayers(x)
- **g**: integer. Index of the Green channel, between 1 and nlayers(x)
- **b**: integer. Index of the Blue channel, between 1 and nlayers(x)
- **zcol**: attribute name or column number in attribute table of the column to be rendered

### Methods (by class)

- RasterStackBrick: stack/brick
- SpatialPixelsDataFrame: SpatialPixelsDataFrame
- ANY: alias for ease of typing
Description

Landsat 8 detail of Franconian Switzerland centered on Poppendorf

Format

"RasterBrick-class" with 5 bands (bands 1 to 5).

Details

Use of this data requires your agreement to the USGS regulations on using Landsat data.

Source

https://earthexplorer.usgs.gov

popupTable

Create HTML strings for popups

Description

These functions are deprecated. Please use leafpop::popupTable, leafpop::popupImage and leafpop::popupGraph instead.

Usage

popupTable(x, zcol, row.numbers = TRUE, feature.id = TRUE)

popupImage(img, src = c("local", "remote"), embed = FALSE, ...)

popupGraph(graphs, type = c("png", "svg", "html"), width = 300, height = 300, ...)

Arguments

x A Spatial* object.

zcol numeric or character vector indicating the columns included in the output popup table. If missing, all columns are displayed.

row.numbers logical whether to include row numbers in the popup table.

feature.id logical whether to add 'Feature ID' entry to popup table.

img A character vector of file path(s) or web-URL(s) to any sort of image file(s).
### renderCubeView

**Description**

Widget render function for use in Shiny

**Usage**

```r
renderCubeView(expr, env = parent.frame(), quoted = FALSE)
```

**Arguments**

- `expr`:
  
  An expression that generates an HTML widget

- `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
renderMapview

Render a mapview widget in shiny

Description

Render a mapview widget in shiny

Usage

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

Arguments

expr An expression that generates an HTML widget
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

renderslideView

Widget render function for use in Shiny

Description

Widget render function for use in Shiny

Usage

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

Arguments

expr An expression that generates an HTML widget
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
show, mapview-method

Method for printing mapview objects (show)

Description

Method for printing mapview objects (show)

Usage

```r
## S4 method for signature 'mapview'
show(object)
```

Arguments

- `object`: a mapview object

slideView

slideView

Description

This function is deprecated. Please use slideview::slideView instead.

Usage

```r
## S4 method for signature 'RasterStackBrick,RasterStackBrick'
slideView(img1, img2,
  label1 = deparse(substitute(img1, env = parent.frame())),
  label2 = deparse(substitute(img2, env = parent.frame())),
  r = 3,
  g = 2, b = 1, na.color = mapviewGetOption("na.color"),
  maxpixels = mapviewGetOption("plainview.maxpixels"), ...)

## S4 method for signature 'RasterLayer,RasterLayer'
slideView(img1, img2,
  label1 = deparse(substitute(img1, env = parent.frame())),
  label2 = deparse(substitute(img2, env = parent.frame())),
  legend = TRUE, col.regions = mapviewGetOption("raster.palette")(256),
  na.color = mapviewGetOption("na.color"),
  maxpixels = mapviewGetOption("plainview.maxpixels"))

## S4 method for signature 'RasterStackBrick,RasterLayer'
slideView(img1, img2,
  label1 = deparse(substitute(img1, env = parent.frame())),
  label2 = deparse(substitute(img2, env = parent.frame())),
  legend = TRUE, r = 3, g = 2, b = 1,
  ...)
```
slideView

col.regions = mapviewgetOption("raster.palette") (256),
na.color = mapviewgetOption("na.color"),
maxpixels = mapviewgetOption("plainview.maxpixels"), ...

## S4 method for signature 'RasterLayer,RasterStackBrick'
slideView(img1, img2,
label1 = deparse(substitute(img1, env = parent.frame())),
label2 = deparse(substitute(img2, env = parent.frame())),
legend = TRUE, r = 3, g = 2, b = 1,
col.regions = mapviewgetOption("raster.palette") (256),
na.color = mapviewgetOption("na.color"),
maxpixels = mapviewgetOption("plainview.maxpixels"), ...

## S4 method for signature 'character,character'
slideView(img1, img2,
label1 = deparse(substitute(img1, env = parent.frame())),
label2 = deparse(substitute(img2, env = parent.frame())))

## S4 method for signature 'ANY'
slideview(...)  

Arguments

- **img1**: a RasterStack/Brick, RasterLayer or path to a .png file
- **img2**: a RasterStack/Brick, RasterLayer or path to a .png file
- **label1**: slider label for img1 (defaults to object name)
- **label2**: slider label for img2 (defaults to object name)
- **r**: integer. Index of the Red channel, between 1 and nlayers(x)
- **g**: integer. Index of the Green channel, between 1 and nlayers(x)
- **b**: integer. Index of the Blue channel, between 1 and nlayers(x)
- **na.color**: the color to be used for NA pixels
- **maxpixels**: integer > 0. Maximum number of cells to use for the plot. If maxpixels < ncell(x), sampleRegular is used before plotting...
- **legend**: whether to plot legends for the two images (ignored for RasterStacks/*Bricks).
- **col.regions**: color (palette). See levelplot for details.
- **color**: the color palette to be used for visualising RasterLayers

Details

Compare two images through interactive swiping overlay
Methods (by class)

- `img1 = RasterLayer, img2 = RasterLayer`: for RasterLayers
- `img1 = RasterStackBrick, img2 = RasterLayer`: for RasterStackBrick, RasterLayer
- `img1 = RasterLayer, img2 = RasterStackBrick`: for RasterLayer, RasterStackBrick
- `img1 = character, img2 = character`: for png files
- `ANY`: alias for ease of typing

---

slideViewOutput  

*Widget output function for use in Shiny*

---

**Description**

Widget output function for use in Shiny

**Usage**

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

**Arguments**

- `outputId`: Output variable to read from
- `width`, `height`: the width and height of the canvas element (see `shinyWidgetOutput`)

---

trails  

*Selected hiking trails in Franconia*

---

**Description**

Selected hiking trails in Franconia

**Format**

`sf feature collection MULTILINESTRING`

**Details**

These hiking trails were downloaded on 06/04/2017 from [https://geoportal.bayern.de/bayernatlas](https://geoportal.bayern.de/bayernatlas)

These data are published by the owner under Creative Commons Namensnennung 3.0 Deutschland, see [https://creativecommons.org/licenses/by/3.0/de/](https://creativecommons.org/licenses/by/3.0/de/) for details.

**Source**

viewExtent  View extent/bbox of spatial objects interactively

Description

This function produces an interactive view of the extent/bbox of the supplied spatial object

Usage

viewExtent(x, map = NULL, popup = NULL, layer.name = NULL,
    alpha.regions = 0.2, label = NULL, ...)

    addExtent(map, data, ...)

Arguments

x       either a Raster*, sf* or Spatial* object
map     a leaflet map the extent should be added to. If NULL standard background layers are created.
popup   a list of HTML strings with the popup contents, usually created from popupTable. See addControl for details.
layer.name the name of the layer to be shown on the map.
alpha.regions opacity of the fills or the raster layer(s).
label   a character vector of labels to be shown on mouseover. See addControl for details.
...     additional arguments passed on to addRectangles
data     either a Raster*, sf* or Spatial* object

Functions

• addExtent: add extent/bbox of spatial/sf objects to a leaflet map

Author(s)

Tim Appelhans

Examples

library(leaflet)

viewExtent(breweries)
viewExtent(franconia) + breweries
leaflet() %>% addProviderTiles("OpenStreetMap") %>% addExtent(breweries)
viewRGB  
**Red-Green-Blue map view of a multi-layered Raster object**

**Description**

Make a Red-Green-Blue plot based on three layers (in a RasterBrick or RasterStack). Three layers (sometimes referred to as "bands" because they may represent different bandwidths in the electromagnetic spectrum) are combined such that they represent the red, green and blue channel. This function can be used to make 'true (or false) color images' from Landsat and other multi-band satellite images. Note, this text is plagiarized, i.e. copied from `plotRGB`.

**Usage**

```r
## S4 method for signature 'RasterStackBrick'
viewRGB(x, r = 3, g = 2, b = 1,
        quantiles = c(0.02, 0.98), map = NULL,
        maxpixels = mapviewGetOption("mapview.maxpixels"),
        map.types = mapviewGetOption("basemaps"),
        na.color = mapviewGetOption("na.color"),
        layer.name = deparse(substitute(x, env = parent.frame())),
        method = c("bilinear", "ngb"), ...)

## S4 method for signature 'stars'
viewRGB(x, r = 3, g = 2, b = 1,
        quantiles = c(0.02, 0.98), map = NULL,
        maxpixels = mapviewGetOption("mapview.maxpixels"),
        map.types = mapviewGetOption("basemaps"),
        na.color = mapviewGetOption("na.color"),
        layer.name = deparse(substitute(x, env = parent.frame())),
        method = c("bilinear", "ngb"), ...)
```

**Arguments**

- `x`: a RasterBrick or RasterStack
- `r`: integer. Index of the Red channel/band, between 1 and nlayers(x)
- `g`: integer. Index of the Green channel/band, between 1 and nlayers(x)
- `b`: integer. Index of the Blue channel/band, between 1 and nlayers(x)
- `quantiles`: the upper and lower quantiles used for color stretching. If set to NULL, no stretching is applied.
- `map`: the map to which the layer should be added
- `maxpixels`: integer > 0. Maximum number of cells to use for the plot. If maxpixels < ncell(x), sampleRegular is used before plotting.
- `na.color`: the color to be used for NA pixels
layer.name  the name of the layer to be shown on the map
method   Method used to compute values for the resampled layer that is passed on to leaflet. mapview does projection on-the-fly to ensure correct display and therefore needs to know how to do this projection. The default is 'bilinear' (bilinear interpolation), which is appropriate for continuous variables. The other option, 'ngb' (nearest neighbor), is useful for categorical variables.

... additional arguments passed on to mapView

Methods (by class)

• stars: stars

Author(s)
Tim Appelhans

Examples

if (interactive()) {
  library(raster)
  library(plainview)

  viewRGB(plainview::poppendorf, 4, 3, 2) # true-color
  viewRGB(plainview::poppendorf, 5, 4, 3) # false-color
}
Index

+Topic package
   mapview-package, 3
    +, 3
    +, mapview, ANY-method (+), 3
    +, mapview, NULL-method (+), 3
    +, mapview, character-method (+), 3
    +, mapview, mapview-method (+), 3

addCircleMarkers, 5
addCircles, 25, 36
addControl, 25, 26, 43
addExtent (viewExtent), 43
addFeatures, 4, 4, 31
addHomeButton, 5, 5, 31
addImageQuery, 6, 6, 31
addListenerOnlyMarkers, 9, 10
addLayersControl, 17
addLegend, 25
addLogo, 7, 7, 31
addMapPane, 26
addMouseCoordinates, 7, 7, 31
addPolygons, 5, 25, 36
addPolylines, 5, 25, 36
addRasterImage, 6, 9, 25, 32, 36
addRectangles, 43
addStarsImage, 8
addStaticLabels, 9
bbox, 5
breweries, 11
brick, 27, 36

colorRampPalette, 32, 33
coords2JSON, 11
coords2JSON, character-method (coords2JSON), 11
coords2JSON, matrix-method (coords2JSON), 11
coords2JSON, numeric-method (coords2JSON), 11

cubeView, 14, 30
cubeview, 30
cubeview (cubeView), 14
cubeViewOutput, 16, 30
data.frame, 27
extent, 5
franconia, 16
garnishMap, 17, 31
highlightOptions, 26

knit_print, 18
knit_print.mapview, 18

labelOptions, 9
latticeView, 18, 18, 30
latticeView (latticeView), 18
leafletSizingPolicy, 25
level.colors, 32
levelplot, 15, 24, 36, 41
Lines, 13
list, 27

mapshot, 19
mapView, 20, 45
mapView (mapView), 20
mapView, ANY-method (mapView), 20
mapView, bbox-method (mapView), 20
mapView, data.frame-method (mapView), 20
mapView, list-method (mapView), 20
mapView, missing-method (mapView), 20
mapView, RasterLayer-method (mapView), 20
mapView, RasterStackBrick-method (mapView), 20
mapView, Satellite-method (mapView), 20
mapView, sf-method (mapView), 20
mapView, sfc-method (mapView), 20
mapView, SpatialGridDataFrame-method (mapView), 20
mapView, SpatialLines-method (mapView), 20
mapView, SpatialLinesDataFrame-method (mapView), 20
mapView, SpatialPixelsDataFrame-method (mapView), 20
mapView, SpatialPoints-method (mapView), 20
mapView, SpatialPointsDataFrame-method (mapView), 20
mapView, SpatialPolygons-method (mapView), 20
mapView, SpatialPolygonsDataFrame-method (mapView), 20
mapView, stars-method (mapView), 20
mapView, XY-method (mapView), 20
mapView, XYM-method (mapView), 20
mapView, XYZ-method (mapView), 20
mapView, XZM-method (mapView), 20
mapView-class, 30
mapView-deprecated, 30
mapview-package, 3
mapviewColors, 31
mapviewGetOption (mapviewOptions), 32
mapviewOptions, 32
mapviewOutput, 34
mapViewPalette (mapviewColors), 31
mapviewPalette (mapviewColors), 31
npts, 35
numeric, 27
options, 34
plainView, 31, 35, 35
plainview, 31
slideViewOutput, 30, 42
SpatialGridDataFrame, 27
SpatialLines, 12, 13, 27
SpatialLinesDataFrame, 13, 27
SpatialPixelsDataFrame, 27, 36
SpatialPoints, 27
SpatialPointsDataFrame, 27
SpatialPolygons, 13, 14, 27
SpatialPolygonsDataFrame, 14, 27
st_bbox, 27
st_sf, 26, 27
st_sfc, 27
stack, 27, 36
stars, 27, 45
sync, 18, 30
sync(latticeView), 18

trails, 42

viewExtent, 43
viewExtent, addExtent (viewExtent), 43
viewRGB, 44
viewRGB, RasterStackBrick-method
(viewRGB), 44
viewRGB, stars-method (viewRGB), 44

webshot, 19, 20