Package 'mapview'

May 8, 2026

Title Interactive Viewing of Spatial Data in R

Version 2.11.4

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 and bounding boxes.

License GPL (>= 3) | file LICENSE

URL https://github.com/r-spatial/mapview, https://r-spatial.github.io/mapview/

BugReports https://github.com/r-spatial/mapview/issues

Depends methods, R (>= 3.6.0)

Imports base64enc, htmltools, htmlwidgets, lattice, leafem, leaflet (>= 2.0.0), leafpop, png, raster (>= 3.6.3), satellite, scales (>= 0.2.5), servr, sf, sp

Suggests knitr, later, leaflet.extras2, leafsync, lwgeom, mapdeck, plainview, poorman, rmarkdown, rstudioapi, s2, stars, tinytest, webshot, webshot2

ByteCompile yes

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.3.2

SystemRequirements GNU make

NeedsCompilation no

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],
Lorenzo Busetto [ctb]

Repository CRAN

Date/Publication 2025-09-08 07:00:18 UTC

Contents

`mapview-package` *mapview: Interactive Viewing of Spatial Data in R*

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 and bounding boxes.

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)

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

Authors:

- Florian Detsch <fdetsch@web.de>
- Christoph Reudenbach <reudenbach@geo.uni-marburg.de>
- Stefan Woellauer <stephan.woellauer@geo.uni-marburg.de>

Other contributors:

- Spaska Forteva <spaska.forteva@geo.uni-marburg.de> [contributor]
- Thomas Nauss <nauss@staff.uni-marburg.de> [contributor]
- Edzer Pebesma [contributor]
- Kenton Russell [contributor]
- Michael Sumner [contributor]
- Jochen Darley <Debugger@jedimasters.de> [contributor]
- Pierre Roudier [contributor]
- Patrick Schratz [contributor]
- Environmental Informatics Marburg [contributor]
- Lorenzo Busetto [contributor]

See Also

Useful links:

- https://github.com/r-spatial/mapview
- https://r-spatial.github.io/mapview/
- Report bugs at https://github.com/r-spatial/mapview/issues

breweries *Selected breweries in Franconia*

Description

Selected breweries in Franconia

Usage

```
breweries
```

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 https://www.bierwandern.de/inhalt/brauereiliste.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.

franconia *Administrative district borders of Franconia*

Description

Administrative district borders of Franconia

Usage

```
franconia
```

Format

`sf feature collection MULTIPOLYGON`

Details

The NUTS_2013_01M_SH.zip archive was downloaded on 23/03/2017 from the now defunct URL "https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts". The current working URL is https://ec.europa.eu/eurostat/web/gisco/geodata/statistical-units/territorial-units-statistics. https://gist.github.com/tim-salabim/2845fa90813fa25c18cf83f9b88cbde0

Source

https://ec.europa.eu/eurostat/web/gisco/geodata

`knit_print.mapview` *Print functions for mapview objects used in knitr*

Description

Print functions for mapview objects used in knitr

Usage

```
## S3 method for class 'mapview'
knit_print(x, ...)
```

Arguments

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

`mapshot` *Save mapview or leaflet map as HTML and/or image using webshot*

Description

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

Usage

```
mapshot(
  x,
  url = NULL,
  file = NULL,
  remove_controls = c("zoomControl", "layersControl", "homeButton", "scaleBar",
    "drawToolbar", "easyButton"),
  ...
)

mapshot2(
  x,
  url = NULL,
  file = NULL,
  remove_controls = c("zoomControl", "layersControl", "homeButton", "scaleBar",
    "drawToolbar", "easyButton", "control"),
  ...
)
```

Arguments

x	mapview or leaflet object (or any other hmtlwidget).
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_controls	character vector of control buttons to be removed from the map when saving to file. Any combination of "zoomControl", "layersControl", "homeButton", "scaleBar", "drawToolbar", "easyButton". If set to NULL nothing will be removed. Ignored if x is not a mapview or leaflet map.
...	Further arguments passed on to saveWidget and/or webshot.

Details

mapshot uses webshot from the webshot package. mapshot2 uses webshot from the webshot2 package.

mapshot can be used to save both leaflet and mapview maps as html or png files or both. In theory, it should also work for any and all other htmlwidgets but has not been tested extensively for other htmlwidgets.

In case you want to save larger maps mapshot is likely to fail. You can try setting selfcontained = FALSE to avoid errors and create a valid local html file.

mapshot2 uses saveWidget and webshot to save maps as .html and/or .png|.jpg files, respectively. webshot assumes a findable installation of some Chrome browser variant on your system. If you see the the following error:

`google-chrome` and `chromium-browser` were not found. Try setting the CHROMOTE_CHROME environment variable or adding one of these executables to your PATH.

it means that find_chrome cannot find a Chrome based browser in your system. Please see https://github.com/rstudio/chromote#specifying-which-browser-to-use for more details.

Functions

- mapshot(): Save mapview or leaflet map as HTML and/or image using webshot
- mapshot2(): Save mapview or leaflet map as HTML and/or image using webshot2

See Also

webshot, saveWidget.

webshot.

Examples

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

  m = mapview(breweries)
  html_fl = tempfile(fileext = ".html")
```

```
  png_fl = tempfile(fileext = ".png")

  ## create standalone .html
  mapshot(m, url = html_fl)
  browseURL(html_fl)

  ## create standalone .png; temporary .html is removed automatically unless
  ## 'remove_url = FALSE' is specified
  mapshot(m, file = png_fl)
  browseURL(png_fl)
  mapshot(m, file = png_fl,
          remove_controls = c("homeButton", "layersControl"))
  browseURL(png_fl)

  ## create .html and .png
  mapshot(m, url = html_fl, file = png_fl)
  browseURL(png_fl)
  browseURL(html_fl)

## End(Not run)

## Not run:
  library(utils)

  m = mapview(breweries)
  html_fl = tempfile(fileext = ".html")
  png_fl = tempfile(fileext = ".png")

  ## create standalone .html
  mapshot2(m, url = html_fl)
  browseURL(html_fl)

  ## create standalone .png; temporary .html is removed automatically unless
  ## 'remove_url = FALSE' is specified
  mapshot2(m, file = png_fl)
  browseURL(png_fl)
  mapshot2(m, file = png_fl,
           remove_controls = c("homeButton", "layersControl"))
  browseURL(png_fl)

  ## create .html and .png
  mapshot2(m, url = html_fl, file = png_fl)
  browseURL(png_fl)
  browseURL(html_fl)

## End(Not run)
```

---

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

```
## S4 method for signature 'RasterLayer'
mapView(
  x,
  map = NULL,
  maxpixels = mapviewGetOption("mapview.maxpixels"),
  col.regions = mapviewGetOption("raster.palette"),
  at = NULL,
  na.color = mapviewGetOption("na.color"),
  use.layer.names = mapviewGetOption("use.layer.names"),
  map.types = mapviewGetOption("basemaps"),
  alpha.regions = 0.8,
  legend = mapviewGetOption("legend"),
  legend.opacity = 1,
  trim = mapviewGetOption("trim"),
  verbose = mapviewGetOption("verbose"),
  layer.name = NULL,
  homebutton = mapviewGetOption("homebutton"),
  native.crs = mapviewGetOption("native.crs"),
  method = mapviewGetOption("method"),
  label = TRUE,
  query.type = mapviewGetOption("query.type"),
  query.digits = mapviewGetOption("query.digits"),
  query.position = mapviewGetOption("query.position"),
  query.prefix = mapviewGetOption("query.prefix"),
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  hide = FALSE,
  ...
)

## S4 method for signature 'stars'
mapView(
  x,
  band = 1,
  map = NULL,
  maxpixels = mapviewGetOption("mapview.maxpixels"),
  col.regions = mapviewGetOption("raster.palette"),
  at = NULL,
  na.color = mapviewGetOption("na.color"),
  use.layer.names = mapviewGetOption("use.layer.names"),
  map.types = mapviewGetOption("basemaps"),
  alpha.regions = 0.8,
  legend = mapviewGetOption("legend"),
```

```
  legend.opacity = 1,
  trim = mapviewGetOption("trim"),
  verbose = mapviewGetOption("verbose"),
  layer.name = NULL,
  homebutton = mapviewGetOption("homebutton"),
  native.crs = mapviewGetOption("native.crs"),
  method = mapviewGetOption("method"),
  label = TRUE,
  query.type = mapviewGetOption("query.type"),
  query.digits = mapviewGetOption("query.digits"),
  query.position = mapviewGetOption("query.position"),
  query.prefix = mapviewGetOption("query.prefix"),
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  pane = "auto",
  hide = FALSE,
  ...
)

## S4 method for signature 'stars_proxy'
mapView(
  x,
  band = 1,
  map = NULL,
  maxpixels = mapviewGetOption("mapview.maxpixels"),
  col.regions = mapviewGetOption("raster.palette"),
  at = NULL,
  na.color = mapviewGetOption("na.color"),
  use.layer.names = mapviewGetOption("use.layer.names"),
  map.types = mapviewGetOption("basemaps"),
  alpha.regions = 0.8,
  legend = mapviewGetOption("legend"),
  legend.opacity = 1,
  trim = mapviewGetOption("trim"),
  verbose = mapviewGetOption("verbose"),
  layer.name = NULL,
  homebutton = mapviewGetOption("homebutton"),
  native.crs = mapviewGetOption("native.crs"),
  method = mapviewGetOption("method"),
  label = TRUE,
  query.type = mapviewGetOption("query.type"),
  query.digits = mapviewGetOption("query.digits"),
  query.position = mapviewGetOption("query.position"),
  query.prefix = mapviewGetOption("query.prefix"),
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  pane = "auto",
  hide = FALSE,
  ...
)
```

```
## S4 method for signature 'SpatRaster'
mapView(
  x,
  band = 1,
  map = NULL,
  maxpixels = mapviewGetOption("mapview.maxpixels"),
  col.regions = mapviewGetOption("raster.palette"),
  at = NULL,
  na.color = mapviewGetOption("na.color"),
  use.layer.names = mapviewGetOption("use.layer.names"),
  map.types = mapviewGetOption("basemaps"),
  alpha.regions = 0.8,
  legend = mapviewGetOption("legend"),
  legend.opacity = 1,
  trim = mapviewGetOption("trim"),
  verbose = mapviewGetOption("verbose"),
  layer.name = NULL,
  homebutton = mapviewGetOption("homebutton"),
  native.crs = mapviewGetOption("native.crs"),
  method = mapviewGetOption("method"),
  label = TRUE,
  query.type = mapviewGetOption("query.type"),
  query.digits = mapviewGetOption("query.digits"),
  query.position = mapviewGetOption("query.position"),
  query.prefix = mapviewGetOption("query.prefix"),
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  pane = "auto",
  hide = FALSE,
  ...
)

## S4 method for signature 'RasterStackBrick'
mapView(
  x,
  map = NULL,
  maxpixels = mapviewGetOption("mapview.maxpixels"),
  col.regions = mapviewGetOption("raster.palette"),
  at = NULL,
  na.color = mapviewGetOption("na.color"),
  use.layer.names = TRUE,
  map.types = mapviewGetOption("basemaps"),
  legend = mapviewGetOption("legend"),
  legend.opacity = 1,
  trim = TRUE,
  verbose = mapviewGetOption("verbose"),
  homebutton = mapviewGetOption("homebutton"),
  method = mapviewGetOption("method"),
```

```
  label = TRUE,
  query.type = c("mousemove", "click"),
  query.digits = mapviewGetOption("query.digits"),
  query.position = mapviewGetOption("query.position"),
  query.prefix = "Layer",
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  hide = FALSE,
  ...
)

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

## S4 method for signature 'sf'
mapView(
  x,
  map = NULL,
  pane = "auto",
  canvas = useCanvas(x),
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  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 = mapviewGetOption("basemaps"),
  verbose = mapviewGetOption("verbose"),
  popup = TRUE,
  layer.name = NULL,
  label = zcol,
  legend = mapviewGetOption("legend"),
  legend.opacity = 1,
  homebutton = mapviewGetOption("homebutton"),
  native.crs = FALSE,
  highlight = mapviewHighlightOptions(x, alpha.regions, alpha, lwd),
  maxpoints = getMaxFeatures(x),
  hide = FALSE,
  ...
)

## S4 method for signature 'SpatVector'
mapView(
  x,
  map = NULL,
  pane = "auto",
  canvas = useCanvas(x),
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  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 = mapviewGetOption("basemaps"),
  verbose = mapviewGetOption("verbose"),
  popup = TRUE,
  layer.name = NULL,
  label = zcol,
  legend = mapviewGetOption("legend"),
  legend.opacity = 1,
  homebutton = mapviewGetOption("homebutton"),
  native.crs = FALSE,
  highlight = mapviewHighlightOptions(x, alpha.regions, alpha, lwd),
  maxpoints = getMaxFeatures(x),
  hide = FALSE,
  ...
)
```

```
## S4 method for signature 'sfc'
mapView(
  x,
  map = NULL,
  pane = "auto",
  canvas = useCanvas(x),
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  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 = mapviewGetOption("basemaps"),
  verbose = mapviewGetOption("verbose"),
  popup = NULL,
  layer.name = deparse(substitute(x, env = parent.frame())),
  label = makeLabels(x),
  legend = mapviewGetOption("legend"),
  legend.opacity = 1,
  homebutton = mapviewGetOption("homebutton"),
  native.crs = FALSE,
  highlight = mapviewHighlightOptions(x, alpha.regions, alpha, lwd),
  maxpoints = getMaxFeatures(x),
  hide = FALSE,
  ...
)

## S4 method for signature 'character'
mapView(
  x,
  map = NULL,
  tms = TRUE,
  color = standardColor(),
  col.regions = standardColRegions(),
  at = NULL,
  na.color = mapviewGetOption("na.color"),
  cex = 6,
  lwd = 2,
  alpha = 0.9,
  alpha.regions = 0.6,
  na.alpha = 0.6,
  map.types = mapviewGetOption("basemaps"),
  verbose = FALSE,
  layer.name = x,
  homebutton = mapviewGetOption("homebutton"),
```

```
  native.crs = FALSE,
  canvas = FALSE,
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  ...
)

## 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, className = "mapview-popup"),
  label,
  crs = NA,
  ...
)

## S4 method for signature 'XY'
mapView(
  x,
  map = NULL,
  pane = "auto",
  canvas = useCanvas(x),
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  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 = mapviewGetOption("basemaps"),
  verbose = mapviewGetOption("verbose"),
  popup = NULL,
  layer.name = deparse(substitute(x, env = parent.frame(1))),
  label = makeLabels(x),
  legend = mapviewGetOption("legend"),
  legend.opacity = 1,
  homebutton = mapviewGetOption("homebutton"),
  native.crs = FALSE,
  highlight = mapviewHighlightOptions(x, alpha.regions, alpha, lwd),
  maxpoints = getMaxFeatures(x),
```

```
  hide = FALSE,
  ...
)

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

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

## 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 'NULL'
mapView(x, ...)

## 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 = 0.9,
  alpha.regions = lapply(x, regionOpacity),
  na.alpha = lapply(x, regionOpacity),
  map.types = mapviewGetOption("basemaps"),
  verbose = mapviewGetOption("verbose"),
  popup = TRUE,
  layer.name = deparse(substitute(x, env = parent.frame()), width.cutoff = 500L),
  label = lapply(x, makeLabels),
  legend = mapviewGetOption("legend"),
```

```
  homebutton = mapviewGetOption("homebutton"),
  native.crs = FALSE,
  hide = FALSE,
  ...
)

## 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"),
  at = NULL,
  na.color = mapviewGetOption("na.color"),
  use.layer.names = FALSE,
  map.types = mapviewGetOption("basemaps"),
  alpha.regions = 0.8,
  legend = mapviewGetOption("legend"),
  legend.opacity = 1,
  trim = TRUE,
  verbose = mapviewGetOption("verbose"),
  layer.name = NULL,
  homebutton = mapviewGetOption("homebutton"),
  native.crs = FALSE,
  method = mapviewGetOption("method"),
  label = TRUE,
  query.type = c("mousemove", "click"),
  query.digits,
  query.position = "topright",
  query.prefix = "Layer",
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  hide = FALSE,
  ...
)

## S4 method for signature 'SpatialGridDataFrame'
mapView(
  x,
  map = NULL,
  zcol = NULL,
  maxpixels = mapviewGetOption("mapview.maxpixels"),
  col.regions = mapviewGetOption("raster.palette"),
  at = NULL,
  na.color = mapviewGetOption("na.color"),
```

```
  use.layer.names = FALSE,
  map.types = mapviewGetOption("basemaps"),
  alpha.regions = 0.8,
  legend = mapviewGetOption("legend"),
  legend.opacity = 1,
  trim = TRUE,
  verbose = mapviewGetOption("verbose"),
  layer.name = NULL,
  homebutton = mapviewGetOption("homebutton"),
  native.crs = FALSE,
  method = mapviewGetOption("method"),
  label = TRUE,
  query.type = c("mousemove", "click"),
  query.digits,
  query.position = "topright",
  query.prefix = "Layer",
  viewer.suppress = mapviewGetOption("viewer.suppress"),
  hide = 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 or stars object or a list of any combination of those. Furthermore, this can also be a data.frame, a numeric vector or a character string pointing to a tile image folder or file on disk. If missing, a blank map will be drawn. A value of NULL will return NULL.
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?
map.types	character specifications for the base maps. see https://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. By default this is the character version of whatever is passed to x. NOTE: This is being passed to underlying leaflet functions as the group argument. So if you use mapview to set up a map and want to refer to a certain layer later on, this is what you should refer to in group.
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	deprecated. Use mapviewOptions(viewer.suppress = TRUE/FALSE) instead.

hide	either a logical, a vector of layer names or a vector of layer indices. See Details for more information on what exactly it does for different raster types.
...	additional arguments passed on to respective functions. See addRasterImage, addCircles, addPolygons, addPolylines for details. Furthermore, you can pass hidden arguments to some methods. See Details for a list of supported hidden arguments.
band	for stars layers, the band number to be plotted.
pane	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 allfeatures are rendered in the same pane and layer order is determined automatically/sequentially.
canvas	whether to use canvas rendering rather than svg. May help performance with larger data. See https://leafletjs.com/index.html#canvas for more information. Only applicable for vector data. The default setting will decide automatically, based on feature complexity.
zcol	attribute name(s) or column number(s) in attribute table of the column(s) to be rendered. See also Details.
burst	whether to show all (TRUE) or only one (FALSE) layer(s). See also Details.
color	color (palette) for points/polygons/lines
cex	attribute name(s) or column number(s) in attribute table of the column(s) to be used for defining the size of circles
lwd	line width
alpha	opacity of lines
na.alpha	opacity of missing values
popup	either logical, character vector or a list of HTML strings with the popup contents, usually created from popupTable. See addControl for details. If FALSE or NULL no popups will be created, if TRUE a table with all feature attributes/columns will be created. If a character vector of column names, the table will only show the respective column entries.
highlight	either FALSE, NULL or a list of styling options for feature highlighting on mouse hover. See highlightOptions for details.
maxpoints	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.
tms	whether the tiles are served as TMS tiles.
y	numeric vector.
type	whether to render the numeric vector x as a point "p" or line "l" plot.
grid	whether to plot a (scatter plot) xy-grid to aid interpretation of the visualisation. Only relevant for the data.frame method.
xcol	the column to be mapped to the x-axis. Only relevant for the data.frame method.
ycol	the column to be mapped to the y-axis. Only relevant for the data.frame method.

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

hide if TRUE, will hide the layer in case of a single RasterLayer and all but the first layer in case of a multilayer RasterStackBrick. If a vector of layer names or indices is supplied, these will be hidden (only applicable for multi-layer RasterStackBricks).

Methods (by class)

- mapView(stars): stars
- mapView(stars_proxy): stars_proxy
- mapView(SpatRaster): SpatRaster
- mapView(RasterStackBrick): stack / brick
- mapView(Satellite): satellite
- mapView(sf): sf
- mapView(SpatVector): SpatVector
- mapView(sfc): st_sfc
- mapView(character): character
- mapView(numeric): numeric
- mapView(data.frame): data.frame
- mapView(XY): st_sfc

- mapView(XYZ): st_sfc
- mapView(XYM): st_sfc
- mapView(XYZM): st_sfc
- mapView(bbox): st_bbox
- mapView(missing): initiate a map without an object
- mapView(`NULL`): initiate a map without an object
- mapView(list): list
- mapview(ANY): alias for ease of typing
- mapView(SpatialPixelsDataFrame): SpatialPixelsDataFrame
- mapView(SpatialGridDataFrame): SpatialGridDataFrame
- mapView(SpatialPointsDataFrame): SpatialPointsDataFrame
- mapView(SpatialPoints): SpatialPoints
- mapView(SpatialPolygonsDataFrame): SpatialPolygonsDataFrame
- mapView(SpatialPolygons): SpatialPolygons
- mapView(SpatialLinesDataFrame): SpatialLinesDataFrame
- mapView(SpatialLines): SpatialLines

Author(s)

Tim Appelhans

Examples

```
## Not run:
  mapview()

  ## simple features ====================================================
  library(sf)

  # sf
  mapview(breweries)
  mapview(franconia)

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

  # sfg / XY - taken from ?sf::st_point
  outer = matrix(c(0,0,10,0,10,10,0,10,0,0),ncol=2, byrow=TRUE)
  hole1 = matrix(c(1,1,1,2,2,2,1,1,1),ncol=2, byrow=TRUE)
  hole2 = matrix(c(5,5,5,6,6,6,5,5,5),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(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(poorman)
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)
```

`mapview-class` *Class mapview*

Description

Class mapview

Slots

`object` the spatial object

`map` the leaflet map object

`mapview-defunct` *Defunct functions in mapview*

Description

These functions have been removed from package `mapview`. See below for information on which package they have been moved to.

Details

- cubeview: This function is defunct, and has been migrated to package 'cubeview'.
- cubeView: This function is defunct, and has been migrated to package 'cubeview'.
- cubeViewOutput: This function is defunct, and has been migrated to package 'cubeview'.
- renderCubeView: This function is defunct, and has been migrated to package 'cubeview'.
- slideview: This function is defunct, and has been migrated to package 'slideview'.
- slideView: This function is defunct, and has been migrated to package 'slideview'.
- slideViewOutput: This function is defunct, and has been migrated to package 'slideview'.
- renderslideView: This function is defunct, and has been migrated to package 'slideview'.
- latticeView: This function is defunct, and has been migrated to package 'leafsync'.
- sync: This function is defunct, and has been migrated to package 'leafsync'.
- plainview: This function is defunct, and has been migrated to package 'plainview'.
- plainView: This function is defunct, and has been migrated to package 'plainview'.
- popupTable: This function is defunct, and has been migrated to package 'leafpop'.
- popupImage: This function is defunct, and has been migrated to package 'leafpop'.
- popupGraph: This function is defunct, and has been migrated to package 'leafpop'.
- addFeatures: This function is defunct, and has been migrated to package 'leafem'.
- garnishMap: This function is defunct, and has been migrated to package 'leafem'.
- addHomeButton: This function is defunct, and has been migrated to package 'leafem'.
- removeHomeButton: This function is defunct, and has been migrated to package 'leafem'.
- addImageQuery: This function is defunct, and has been migrated to package 'leafem'.
- addLogo: This function is defunct, and has been migrated to package 'leafem'.
- addMouseCoordinates: This function is defunct, and has been migrated to package 'leafem'.
- removeMouseCoordinates: This function is defunct, and has been migrated to package 'leafem'.
- addStaticLabels: This function is defunct, and has been migrated to package 'leafem'.
- addExtent: This function is defunct, and has been migrated to package 'leafem'.
- addStarsImage: This function is defunct, and 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

---

mapviewOptions *Global options for the mapview package*

Description

To permanently set any of these options, you can add them to your 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,
  basemaps.color.shuffle,
  raster.palette,
  vector.palette,
  verbose,
  na.color,
  legend,
  legend.opacity,
  legend.pos,
  layers.control.pos,
  leafletWidth,
  leafletHeight,
  viewer.suppress,
  homebutton,
  homebutton.pos,
  native.crs,
  raster.size,
  mapview.maxpixels,
  plainview.maxpixels,
  use.layer.names,
  trim,
  method,
  query.type,
  query.digits,
  query.position,
  query.prefix,
  maxpoints,
  maxpolygons,
  maxlines,
  pane,
  cex,
  alpha,
  default = FALSE,
  console = TRUE,
  watch = FALSE,
  fgb,
  georaster
)

mapviewGetOption(param)
```

Arguments

platform	character. The rendering platform to be used. Current options are "leaflet", "mapdeck", and "leafgl".

basemaps character. The basemaps to be used for rendering data. See https://leaflet-extras.github.io/leaflet-providers/preview/ for possible values

basemaps.color.shuffle
logical. Should basemaps order be changed to enhance contrast based on layer coloring. Set to FALSE if you supply custom basemaps or want to ensure that "CartoDB.Positron" is always the default.

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.opacity opacity of the legend.

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".

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

viewer.suppress
whether to render the map in the browser (TRUE) or the RStudio viewer (FALSE).

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

homebutton.pos character. Where should the homebutton(s) be placed? One of "topleft", "topright", "bottomleft", "bottomright".

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).

raster.size numeric. see the maxBytes argument in addRasterImage

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 10000000. Set this higher if you have a potent machine or are patient enough to wait a little.

use.layer.names
: whether to use layer names when plotting raster layers.

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

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.

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`.

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.

pane
: 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 automatically/sequentially.

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

alpha
: opacity of lines.

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

console
: logical. Should the options be printed to the console

watch	whether to watch a certain environment and automatically render changes to the list of spatial data in that environment. See mapviewWatcher for details.
fgb	if set to TRUE mapview will not use 'classical' leaflet/htmlwidgets rendering (which embeds data directly in the html) but leverage the speed of a file format called flatgeobuf (hence, fgb). This has the added benefit that data is being streamed onto the map, which makes for a pleasant user experience. It should also help to visualise larger data sets due to a reduced memory footprint. A note of warning, data will be attached to the html via a <src=...> call which means that the html is not selfcontained anymore (so it cannot be used without an accompanying folder).
georaster	whether to use addGeoRaster instead of addRasterImage. If set to TRUE raster image visualisation will be more performant for large raster data, but given the nearest neighbor resampling results may be slightly distorted.
param	character. parameter(s) to be queried.

Value

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

Functions

- mapviewGetOption(): query mapviewOptions parameters.

Author(s)

Tim Appelhans

See Also

rasterOptions, options

Examples

```
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

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

Arguments

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

mapviewWatcher *Start and/or stop automagic mapviewing of spatial objects in your workspace.*

Description

Use these functions to enable automatic viewing of all spatial objects currently available in env. mapviewWatcher uses later to set up a watcher function that continuously monitors env for spatial objects and refreshes the viewer/browser in case the list of spatial objects changes.

startWatching and stopWatching are convenience functions to start and stop watching, respectively.

Usage

```
mapviewWatcher(env = .GlobalEnv, ...)

startWatching(env = .GlobalEnv, ...)

stopWatching(env = .GlobalEnv, ...)
```

Arguments

env	the environment that is being watched (default is .GlobalEnv).
...	currently not used.

Details

mapviewWatcher uses identical and hence will redraw even if e.g. the attributes of a spatial object are changed only slightly. By default mapviewWatcher watches the .GlobalEnv but this can be changed to another environment. Whether watching is turned on is controlled by mapviewGetOption("watch"). In order to enable watching it needs to be set to mapviewOptions(watch = TRUE) (default is FALSE) and the watcher needs to be initiated by calling mapviewWatcher() once. To switch watching off it is sufficient to set mapviewOptions(watch = FALSE).

Functions

- startWatching(): start watching
- stopWatching(): stop watching

Examples

```
if (interactive()) {
  library(mapview)

  ## start the watcher
  mapview::startWatching()

  ## load some data and watch the automatic visualisation
  fran = mapview::franconia
  brew = mapview::breweries

  ## stop the watcher
  mapview::stopWatching()

  ## loading or removing things now will not trigger a view update
  rm(brew)
  trls = mapview::trails

  ## re-starting the viewer will re-draw whatever is currently available
  mapview::startWatching()

  ## watcher can also be stopped via mapviewOptions
  mapviewOptions(watch = FALSE)

  rm(trls)

}
```

---

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, by_feature = 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, by_feature = TRUE)
npts(sf::st_geometry(franconia[1, ])) # first polygon

npts(breweries) # is the same as
nrow(breweries)
```

---

ops *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

...

mapview | mapview provides a slider in the middle to compare two maps.

mapview | NULL returns the LHS map

NULL | mapview returns the RHS map

Usage

```
## S4 method for signature 'mapview,mapview'
e1 + e2

## S4 method for signature 'mapview,ANY'
e1 + e2

## S4 method for signature 'mapview,NULL'
e1 + e2
```

```
## S4 method for signature 'mapview,character'
e1 + e2

## S4 method for signature 'mapview,NULL'
e1 | e2

## S4 method for signature 'NULL,mapview'
e1 | e2
```

Arguments

e1	a leaflet or mapview map, or NULL.
e2	a leaflet or mapview map, or NULL.

Examples

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

### add two mapview objects
m1 + m2

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

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

### add two mapview objects
m1 | m2
```

---

print,mapview-method *Method for printing mapview objects*

Description

Method for printing mapview objects

Usage

```
## S4 method for signature 'mapview'
print(x)
```

Arguments

x	a mapview object

`removeMapJunk` *Delete elements from a map.*

Description

Delete elements from a map.

Usage

```
removeMapJunk(map, junk = NULL)
```

Arguments

map	the map from which to remove elements.
junk	a charcter vector of elements to remove. If NULL (the default), nothing is removed and the map is returned as is. See Details for a list of currently supported elements.

Details

Currently supports removal of

- "zoomControl"
- "layersControl"
- "homeButton"
- "scaleBar"
- "drawToolbar"
- "easyButton"

This is mainly useful when taking a static screenshot of a map.

Examples

```
if (interactive()) {
  library(mapview)

  map = mapview(franconia)

  removeMapJunk(map, "zoomControl")
}
```

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

show,mapview-method *Method for printing mapview objects (show)*

Description

Method for printing mapview objects (show)

Usage

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

Arguments

object	a mapview object

trails *Selected hiking trails in Franconia*

Description

Selected hiking trails in Franconia

Usage

```
trails
```

Format

`sf feature collection MULTILINESTRING`

Details

These hiking trails were downloaded on 06/04/2017 from https://geoportal.bayern.de/bayernatlas These data are published by the owner under Creative Commons Namensnennung 4.0 Deed, see https://creativecommons.org/licenses/by/4.0/deed.de for details.

Source

Datenquelle: Bayerische Vermessungsverwaltung - www.geodaten.bayern.de https://geodaten.bayern.de/opengeodata/OpenDataDetail.html?pn=bvv_wanderwege

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,
  ...
)
```

Arguments

x	either a Raster*, sf* or Spatial* object
map	a leaflet or mapview 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.
...	Arguments passed on to leaflet::addRectangles
	layerId the layer id
	data the data object from which the argument values are derived; by default, it is the data object provided to leaflet() initially, but can be overridden
	group the name of the group the newly created layers should belong to (for clearGroup() and addLayersControl() purposes). Human-friendly group names are permitted–they need not be short, identifier-style names. Any number of layers and even different types of layers (e.g., markers and polygons) can share the same group name.
	options a list of extra options for tile layers, popups, paths (circles, rectangles, polygons, ...), or other map elements
	popupOptions A Vector of popupOptions() to provide popups
	labelOptions A Vector of labelOptions() to provide label options for each label. Default NULL
	stroke whether to draw stroke along the path (e.g., the borders of polygons or circles)
	color stroke color
	weight stroke width in pixels
	opacity stroke opacity (or layer opacity for tile layers)
	fill whether to fill the path with color (e.g., filling on polygons or circles)
	fillColor fill color
	fillOpacity fill opacity
	dashArray a string that defines the stroke dash pattern
	highlightOptions Options for highlighting the shape on mouse over.
	smoothFactor how much to simplify the polyline on each zoom level (more means better performance and less accurate representation)
	noClip whether to disable polyline clipping
	lng1,lat1,lng2,lat2 latitudes and longitudes of the south-west and north-east corners of rectangles

Author(s)

Tim Appelhans

Examples

```
library(leaflet)

viewExtent(breweries)
viewExtent(franconia) + breweries
mapview(franconia) %>% leafem::addExtent(franconia, fillColor = "yellow")
leaflet() %>% addProviderTiles("OpenStreetMap") %>% leafem::addExtent(breweries)
leaflet() %>% addProviderTiles("OpenStreetMap") %>% leafem::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, 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 `raster::plotRGB()`.

Usage

```
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 = NULL,
  method = c("bilinear", "ngb"),
  ...
)
```

Arguments

x	a RasterBrick, 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.
map.types	character spcifications for the base maps. see https://leaflet-extras.github.io/leaflet-providers/preview/ for available options.
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

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