

# Package ‘plainview’

April 11, 2022

**Title** Plot Raster Images Interactively on a Plain HTML Canvas

**Version** 0.2.0

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

**Description** Provides methods for plotting potentially large (raster) images interactively on a plain HTML canvas. In contrast to package 'mapview' data are plotted without background map, but data can be projected to any spatial coordinate reference system. Supports plotting of classes 'RasterLayer', 'RasterStack', 'RasterBrick' (from package 'raster') as well as 'png' files located on disk. Interactivity includes zooming, panning, and mouse location information. In case of multi-layer 'RasterStacks' or 'RasterBricks', RGB image plots are created (similar to 'raster::plotRGB' - but interactive).

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 2.10), methods

**Imports** htmltools, htmlwidgets, lattice, png, raster, viridisLite

**Suggests** shiny, sf, sp

**LazyData** true

**RoxygenNote** 7.1.2

**URL** <https://r-spatial.github.io/plainview/>

**NeedsCompilation** no

**Author** Tim Appelhans [cre, aut],  
Stefan Woellauer [aut]

**Repository** CRAN

**Date/Publication** 2022-04-11 17:20:02 UTC

## R topics documented:

plainview-package	2
plainView	2
plainViewOutput	4
poppendorf	5

**Index****6**


---

plainview-package	<i>Plot Raster Images Interactively on a Plain HTML Canvas</i>
-------------------	--

---

**Description**

Plot Raster Images Interactively on a Plain HTML Canvas

**Details**

Provides methods for plotting potentially large (raster) images interactively on a plain HTML canvas. Supports plotting of RasterLayer, RasterStack, RasterBrick (from package raster) as well as png files located on disk. Interactivity includes zooming, panning, and mouse location information. In case of multi-layer RasterStacks or RasterBricks, RBG image plots are created (similar to raster::plotRGB - but interactive).

**Author(s)**

Tim Appelhans, Stephan Woellauer *Maintainer*: Tim Appelhans <tim.appelhans@gmail.com>

---

plainView	<i>View raster objects interactively without background map but in any CRS</i>
-----------	--

---

**Description**

this function produces an interactive view of the specified raster object(s) on a plain grey background but for any CRS.

**Usage**

```
## S4 method for signature 'RasterLayer'
plainView(
  x,
  maxpixels = 1e+08,
  col.regions = viridisLite::inferno,
  at,
  na.color = "#BEBEBE",
  legend = TRUE,
  verbose = FALSE,
  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 = "#BEBEBE",
  maxpixels = 1e+08,
  layer.name = deparse(substitute(x, env = parent.frame())),
  ...
)

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

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

```

### Arguments

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

### Details

If the raster object is not in memory (i.e. if `raster::inMemory` is FALSE) and argument `gdal` is set to TRUE (default) `gdal_translate` is used to translate the raster object to a png file to be rendered in the viewer/browser. This is fast for large rasters. In this case, argument `maxpixels` is not used, instead the image is rendered in original resolution. However, this means that `RasterLayers` will be shown in greyscale. If you want to set a color palette manually, use `gdal = FALSE` and (optionally provide) `col.regions`.

For plainView there are a few keyboard shortcuts defined:

- plus/minus - zoom in/out
- space - toggle antialiasing
- esc - zoom to layer extent
- enter - set zoom to 1
- ctrl - increase panning speed by 10

### Methods (by class)

- RasterStackBrick: [stack](#) / [brick](#)
- SpatialPixelsDataFrame: [SpatialPixelsDataFrame](#)
- ANY: alias for ease of typing

### Author(s)

Stephan Woellauer

Tim Appelhans

### Examples

```
if (interactive()) {

  # RasterLayer
  plainView(poppendorf[[4]])

  # RasterStack
  plainview(poppendorf, r = 4, g = 3, b = 2) # true color
  plainview(poppendorf, r = 5, g = 4, b = 3) # false color

}
```

---

plainViewOutput	<i>Widget output/render function for use in Shiny</i>
-----------------	---

---

### Description

Widget output/render function for use in Shiny

### Usage

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

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

**Arguments**

outputId	Output variable to read from
width, height	the width and height of the map (see <a href="#">shinyWidgetOutput</a> )
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

**Examples**

```

if (interactive()) {
  library(shiny)

  plt = plainView(poppendorf[[4]])

  ui = fluidPage(
    plainViewOutput("plot")
  )

  server = function(input, output, session) {
    output$plot <- renderPlainView(plt)
  }

  shinyApp(ui, server)
}

```

---

poppendorf

*Landsat 8 detail of Franconian Switzerland centered on Poppendorf*


---

**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>

# Index

\* **package**

plainview-package, [2](#)

brick, [4](#)

levelplot, [3](#)

plainView, [2](#)

plainview (plainView), [2](#)

plainview, ANY-method (plainView), [2](#)

plainView, RasterLayer-method

(plainView), [2](#)

plainView, RasterStackBrick-method

(plainView), [2](#)

plainView, SpatialPixelsDataFrame-method

(plainView), [2](#)

plainview-package, [2](#)

plainViewOutput, [4](#)

poppendorf, [5](#)

raster, [3](#)

renderPlainView (plainViewOutput), [4](#)

shinyWidgetOutput, [5](#)

SpatialPixelsDataFrame, [4](#)

stack, [4](#)