Package 'cubeview'

October 12, 2022

Type Package

Title View 3D Raster Cubes Interactively

Version 0.2.0

Date 2019-09-23

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

Description Creates a 3D data cube view of a RasterStack/Brick, typically a collection/array of RasterLayers (along z-axis) with the same geographical extent (x and y dimensions) and resolution, provided by package 'raster'. Slices through each dimension (x/y/z), freely adjustable in location, are mapped to the visible sides of the cube. The cube can be freely rotated. Zooming and panning can be used to focus on different areas of the cube.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 2.10)

Imports base64enc, htmltools, htmlwidgets, lattice, raster, stars, viridisLite

Suggests methods, shiny

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Tim Appelhans [cre, aut], Stefan Woellauer [aut], three.js authors [ctb, cph] (three.js library)

Repository CRAN

Date/Publication 2019-09-24 07:20:02 UTC

R topics documented:

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

Description

Creates a 3D data cube view of a RasterStack/Brick. Slices through each dimension (x/y/z) are mapped to the visible sides of the cube. The cube can be freely rotated. Zooming and panning can be used to focus on different areas of the cube.

See Details for information on how to control the location of the slices and all other available keyborad and mouse guestures to control the cube.

Usage

```
cubeview(x, ...)

## S3 method for class 'character'
cubeview(x, at, col.regions = viridisLite::inferno,
  na.color = "#BEBEBE", legend = TRUE, ...)

## S3 method for class 'stars'
cubeview(x, at, col.regions = viridisLite::inferno,
  na.color = "#BEBEBE", legend = TRUE, ...)

## S3 method for class 'Raster'
cubeview(x, at, col.regions = viridisLite::inferno,
  na.color = "#BEBEBE", legend = TRUE, ...)

cubeView(x, ...)
```

Arguments

x	a file name, stars object, RasterStack or RasterBrick
...	additional arguments passed on to read_stars.
at	the breakpoints used for the visualisation. See levelplot for details.
col.regions	either a palette function or a vector of colors to be used for palette generation.
na.color	color for missing values.
legend	logical. Whether to plot a legend.

Details

The location of the slices can be controlled by keys:
x-axis: LEFT / RIGHT arrow key
y-axis: DOWN / UP arrow key
z-axis: PAGE_DOWN / PAGE_UP key

Other controls:
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.
Press space bar to show/hide slice position guides.

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 may not always recognised within RStudio on Windows. In this case open the view in a web-browser (RStudio button at viewer: "show in new window").

Author(s)

Stephan Woellauer and Tim Appelhans

Examples

```
if (interactive()) {
  library(raster)
  library(stars)

  ## directly from file
  kili_data <- system.file("extdata", "kiliNDVI.tif", package = "cubeview")
  cubeview(kili_data)

  ## stars object
  kili_strs = read_stars(kili_data)
  cubeview(kili_strs)

  ## rsater stack (also works with brick)
  kili_rstr <- stack(kili_data)
  cubeview(kili_rstr)

  ## use different color palette and set breaks
  clr <- viridisLite::viridis
  cubeview(kili_data, at = seq(-0.15, 0.95, 0.1), col.regions = clr)
}
```

`cubeViewOutput` *Widget output/render function for use in Shiny*

Description

Widget output/render function for use in Shiny

Usage

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

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

Arguments

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

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

  cube = cubeView(kiliNDVI)

  ui = fluidPage(
    cubeViewOutput("cube", width = 300, height = 300)
  )

  server = function(input, output, session) {
    output$cube <- renderCubeView(cube)
  }

  shinyApp(ui, server)

}
```

Index