Package 'graph3d'

November 12, 2020

Type Package

Title A Wrapper of the JavaScript Library 'vis-graph3d'

Version 0.2.0

Date 2020-11-12

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Create interactive visualization charts to draw data in three dimensional graphs. The graphs can be included in Shiny apps and R markdown documents, or viewed from the R console and 'RStudio' Viewer. Based on the 'vis.js' Graph3d module and the 'htmlwidgets' R package.

License GPL-3

Imports htmlwidgets, lazyeval

Suggests shiny, viridisLite

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL https://github.com/stla/graph3d

BugReports https://github.com/stla/graph3d/issues

NeedsCompilation no

Author Stéphane Laurent [aut, cre] (R interface),
B. V. Almende [aut, cph] (vis.js library),
vis.js contributors [aut, cph]

Repository CRAN

Date/Publication 2020-11-12 20:00:02 UTC

R topics documented:

graph3d *3D chart*

Description

Generate an interactive 3D chart.

Usage

```
graph3d(
  data = NULL,
  x = ~x,
  y = ~y,
  z = ~z,
  frame = NULL,
  style = NULL,
  type = "surface",
  surfaceColors = c("#FF0000", "#FFF000", "#00FF00", "#68E8FB", "#000FFF"),
  dataColor = NULL,
  xBarWidth = NULL,
  yBarWidth = NULL,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  xValueLabel = NULL,
  yValueLabel = NULL,
  zValueLabel = NULL,
  width = "100%",
  height = "100%",
  backgroundColor = NULL,
  showPerspective = TRUE,
  showGrid = TRUE,
  showShadow = FALSE,
  showXAxis = TRUE,
  showYAxis = TRUE,
  showZAxis = TRUE,
  axisColor = NULL,
  axisFontSize = 30,
  gridColor = NULL,
  keepAspectRatio = TRUE,
  verticalRatio = 0.5,
  tooltip = TRUE,
  tooltipDelay = NULL,
  tooltipStyle = NULL,
  showLegend = TRUE,
  legendLabel = NULL,
  cameraPosition = list(horizontal = 1, vertical = 0.5, distance = 2.8),
```

```
  xCenter = NULL,
  yCenter = NULL,
  xMin = NULL,
  xMax = NULL,
  yMin = NULL,
  yMax = NULL,
  zMin = NULL,
  zMax = NULL,
  xStep = NULL,
  yStep = NULL,
  zStep = NULL,
  showAnimationControls = TRUE,
  animationInterval = 100,
  animationPreload = TRUE,
  frameLabel = NULL,
  onclick = NULL,
  elementId = NULL
)
```

Arguments

`data`	dataframe containing the data for the chart; if not `NULL`, the variables passed to `x`, `y`, `z`, `frame` and `style` are searched among the columns of `data`
`x`	a right-sided formula giving the variable for the locations of the points on the x-axis; required
`y`	a right-sided formula giving the variable for the locations of the points on the y-axis; required
`z`	a right-sided formula giving the variable for the locations of the points on the z-axis; required
`frame`	a right-sided formula giving the variable for the frames of the animation; optional
`style`	a right-sided formula required for `type="dot-color"` and `type="dot-size"`; the variable given by this formula can be a numeric vector for the data value appearing in the legend, or a list of style properties; see the examples
`type`	the type of the chart, one of `"bar"`, `"bar-color"`, `"bar-size"`, `"dot"`, `"dot-line"`, `"dot-color"`, `"dot-size"`, `"line"`, `"grid"`, or `"surface"`
`surfaceColors`	a vector of colors for `type="surface"`, or a list of the form `list(hue = list(start=-360,end=360,sat` see the vis-graph3d documentation for more information
`dataColor`	a string or a list; see the `type="line"` example and the vis-graph3d documentation
`xBarWidth, yBarWidth`	the widths of bars in x and y directions for `type="bar"` and `type="bar-color"`; by default, the width is equal to the smallest distance between the data points
`xlab`	string, the label on the x-axis
`ylab`	string, the label on the y-axis

zlab	string, the label on the z-axis
xValueLabel	JavaScript function for custom formatting of the labels along the x-axis, for example `JS("function(x){return (x * 100) + '%'}")`
yValueLabel	same as `xValueLabel` for the y-axis
zValueLabel	same as `xValueLabel` for the z-axis
width, height	the dimensions of the chart given as strings, in pixels (e.g. `"400px"`) or percentages (e.g. `"80%"`)
backgroundColor	the background color of the chart, either a string giving a HTML color (like `"red"` or `"#00CC00"`), or a list of the form `list(fill="black",stroke="yellow",strokeWidth=3)`; `fill` is the chart fill color, `stroke` is the color of the chart border, and `strokeWidth` is the border width in pixels
showPerspective	logical; if `TRUE`, the graph is drawn in perspective: points and lines which are further away are drawn smaller
showGrid	logical; if `TRUE`, grid lines are drawn in the x-y surface
showShadow	logical, whether to show shadow on the graph
showXAxis	logical; if `TRUE`, x-axis and x-axis labels are drawn
showYAxis	logical; if `TRUE`, y-axis and y-axis labels are drawn
showZAxis	logical; if `TRUE`, z-axis and z-axis labels are drawn
axisColor	a HTML color given as a string, the color of the axis lines and the text along the axes
axisFontSize	a positive number, the font size of the axes labels
gridColor	a HTML color given as a string, the color of the grid lines
keepAspectRatio	logical; if `TRUE`, the x-axis and the y-axis keep their aspect ratio; if `FALSE`, the axes are scaled such that they both have the same, maximum width
verticalRatio	value between 0.1 and 1 which scales the vertical size of the graph; when `keepAspectRatio=FALSE` and `verticalRatio=1`, the graph will be a cube
tooltip	logical, whether to see the tooltips, or a JavaScript function to customize the tooltips; see the barplot example
tooltipDelay	a number, the delay time in ms for the tooltip to appear when the mouse cursor hovers over an x-y grid tile
tooltipStyle	a list of tooltip style properties; see the vis-graph3d documentation
showLegend	logical, whether to see the legend if the graph type supports it
legendLabel	a string, the label of the legend
cameraPosition	a list with three fields to set the initial rotation and position if the camera: `horizontal`, a value in radians, `vertical`, a value in radians between 0 and pi/2, and `distance`, the distance between 0.71 and 5 from the camera to the center of the graph
xCenter	a string giving the horizontal center position of the graph as a percentage (like `"50%"`) or in pixels (like `"100px"`); default to `"55%"`

yCenter	same as xCenter for the vertical center position of the graph; default to "45%"
xMin	minimum value for the x-axis; if not set, the smallest value of x is used
xMax	maximum value for the x-axis; if not set, the largest value of x is used
yMin	minimum value for the y-axis; if not set, the smallest value of y is used
yMax	maximum value for the y-axis; if not set, the largest value of y is used
zMin	minimum value for the z-axis; if not set, the smallest value of z is used
zMax	maximum value for the z-axis; if not set, the largest value of z is used
xStep	a number, the step size for the grid on the x-axis
yStep	a number, the step size for the grid on the y-axis
zStep	a number, the step size for the grid on the z-axis
showAnimationControls	logical, only applicable when the graph contains an animation (i.e. frame is not NULL), whether to show the animation controls (buttons previous, start/stop, next, and a slider)
animationInterval	a number, the animation interval in milliseconds; default to 1000
animationPreload	logical; if FALSE, the animation frames are loaded as soon as they are requested; if TRUE, the animation frames are automatically loaded in the background
frameLabel	string, the label for the animation slider
onclick	a JavaScript function to handle the click event on a point; see the vis-graph3d documentation and the second example in graph3d-shiny
elementId	an id for the widget

Details

See the vis-graph3d documentation.

Examples

```
# 3d bar plot ####
dat <- data.frame(x = c(1,1,2,2), y = c(1,2,1,2), z = c(1,2,3,4))
graph3d(dat, type = "bar", zMin = 0)
# change bar widths
graph3d(dat, type = "bar", zMin = 0, xBarWidth = 0.3, yBarWidth = 0.3)
# with custom tooltips
graph3d(dat, type = "bar", zMin = 0,
        tooltip = JS(c("function(xyz){",
                       "  var x = 'X: ' + xyz.x.toFixed(2);",
                       "  var y = 'Y: ' + xyz.y.toFixed(2);",
                       "  var z = 'Z: ' + xyz.z.toFixed(2);",
                       "  return  x + '<br/>' + y + '<br/>' + z;",
                       "}"))
)

# bivariate Gaussian density ####
```

```
dat <- expand.grid(
  x = seq(-4,4,length.out=100),
  y = seq(-4,4,length.out=100)
)
dat <- transform(dat, density = dnorm(x)*dnorm(y))
graph3d(dat, z = ~density, keepAspectRatio = FALSE, verticalRatio = 1)

# animation ####
f <- function(x, y) sin(x/50) * cos(y/50) * 50 + 50
t_ <- seq(0, 2*pi, length.out = 90)[-90]
x_ <- y_ <- seq(0, 314, length.out = 50)
dat <- expand.grid(x = x_, y = y_, t = t_)
dat <- transform(dat, z = f(x*cos(t) - y*sin(t), x*sin(t) + y*cos(t)))
graph3d(dat, frame = ~t, tooltip = FALSE)

# scatterplot ####
dat <- iris
dat$style <- I(lapply(iris$Species, function(x){
  switch(as.character(x),
         setosa     = list(fill="red",   stroke="#'000"),
         versicolor = list(fill="green", stroke="#'000"),
         virginica  = list(fill="blue",  stroke="#'000"))
}))
graph3d(dat, x = ~Sepal.Length, y = ~Sepal.Width, z = ~Petal.Length,
        style = ~style, type = "dot-color", showLegend = FALSE)

# line ####
t_ <- seq(0, 2*pi, length.out = 200)
dat <- data.frame(
  x = cos(t_),
  y = sin(t_),
  z = 2 * cos(3*t_)
)
graph3d(dat, type = "line", dataColor = list(strokeWidth = 5, stroke = "red"),
        verticalRatio = 1)

# a complex function ####
dat <- expand.grid(
  x = seq(-1, 1, length.out = 100),
  y = seq(-1, 1, length.out = 100)
)
dat <- transform(dat, sine = sin(x + 1i*y))
dat <- transform(dat, modulus = Mod(sine), phase = Arg(sine))
graph3d(dat, z = ~modulus, style = ~phase, type = "dot-color",
        legendLabel = "phase")
```

---

graph3d-imports *Objects imported from other packages*

Description

These objects are imported from other packages. Follow the links to their documentation: JS, saveWidget.

---

graph3d-shiny *Shiny bindings for graph3d*

Description

Output and render functions for using graph3d within Shiny applications and interactive Rmd documents.

Usage

```
graph3dOutput(outputId, width = "100%", height = "400px")

renderGraph3d(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	dimensions, must be valid CSS units (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended
expr	an expression that generates a graph3d HTML widget
env	the environment in which to evaluate expr
quoted	logical, whether expr is a quoted expression (with quote()); this is useful if you want to save an expression in a variable

Examples

```
if(interactive()) {

# 'surfaceColors' example ####

library(shiny)
library(viridisLite)
library(graph3d)

x <- y <- seq(-10, 10, length.out = 100)
dat <- expand.grid(x = x, y = y)
f <- function(x, y){
  r <- sqrt(x^2+y^2)
  10 * ifelse(r == 0, 1, sin(r)/r)
}
dat <- transform(dat, z = f(x, y))

ui <- fluidPage(
```

```
  br(),
  fluidRow(
    column(
      width = 2,
      radioButtons("colors", "Colors",
                   c("viridis", "inferno", "magma", "plasma", "cividis"))
    ),
    column(
      width = 10,
      graph3dOutput("mygraph", height = "550px")
    )
  )
)

server <- function(input, output, session){

  Colors <- reactive({
    colors <- switch(
      input$colors,
      viridis = viridis(5),
      inferno = inferno(5),
      magma = magma(5),
      plasma = plasma(5),
      cividis = cividis(5)
    )
    substring(colors, 1L, 7L)
  })

  output[["mygraph"]] <- renderGraph3d({
    graph3d(dat, surfaceColors = Colors(), showLegend = FALSE)
  })

}

shinyApp(ui, server)

}

if(interactive()) {

# 'onclick' example ####

library(shiny)
library(graph3d)

dat <- data.frame(x = rnorm(30), y = rnorm(30), z = rnorm(30))

onclick <- c(
  "function(point){",
  "  Shiny.setInputValue('point', point);",
  "}"
)
```

```
ui <- fluidPage(
  br(),
  fluidRow(
    column(
      width = 4,
      h4("You clicked:"),
      verbatimTextOutput("pointClicked")
    ),
    column(
      width = 8,
      graph3dOutput("mygraph", height = "550px")
    )
  )
)

server <- function(input, output, session){

  output[["mygraph"]] <- renderGraph3d({
    graph3d(dat, type = "dot", width = "550px", height = "550px",
            onclick = JS(onclick), tooltip = FALSE)
  })

  output[["pointClicked"]] <- renderPrint({
    input[["point"]]
  })

}

shinyApp(ui, server)

}
```

Index