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 .................................................................  2
  graph3d-imports ......................................................  6
  graph3d-shiny ........................................................  7

Index 10
Description

Generate an interactive 3D chart.

Usage

```r
graph3d(
  data = NULL,
  x = ~x,
  y = ~y,
  z = ~z,
  frame = NULL,
  style = NULL,
  type = "surface",
  surfaceColors = c("#FF0000", "#FF0000", "#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),
)```
graph3d

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,saturation=50,brightness=100,colorStops=8)); 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
graph3d

zlab
xValueLabel
yValueLabel
zValueLabel
width, height
backgroundColor
showPerspective
showGrid
showShadow
showXAxis
showYAxis
showZAxis
axisColor
axisFontSize
gridColor
keepAspectRatio
verticalRatio
tooltip
tooltipDelay
tooltipStyle
showLegend
legendLabel
cameraPosition
xCenter
graph3d

In this document, the following parameters are defined:

- **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**

```r
# 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 ####
```

The `graph3d` function allows for the creation of three-dimensional graphs with customizable parameters such as axis limits, grid step sizes, and animation controls. The `onclick` parameter enables the handling of click events on points in the graph, demonstrating the flexibility of the `graph3d` function in creating interactive visualizations.
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="#000000"),
    versicolor = list(fill="green", stroke="#000000"),
    virginica = list(fill="blue", stroke="#000000"))
}))
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")
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**

```r
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**

```r
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

graph3d, 2, 7
graph3d-imports, 6
graph3d-shiny, 7
graph3dOutput (graph3d-shiny), 7

JS, 7
JS (graph3d-imports), 6

renderGraph3d (graph3d-shiny), 7

saveWidget, 7
saveWidget (graph3d-imports), 6