

Package ‘scatterPlotMatrix’

October 21, 2021

Title 'Htmlwidget' for a Scatter Plot Matrix & Multiple Scatter Plot

Version 0.1.0

Description Create a scatter plot matrix, using 'htmlwidgets' package and 'd3.js'.

URL <https://ifpen.gitlabfr.com/detocs/scatterplotmatrix>

BugReports <https://ifpen.gitlabfr.com/detocs/scatterplotmatrix/-/issues>

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 7.1.1

Imports htmlwidgets

Suggests testthat, shiny, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Mike Bostock [aut, cph] (d3.js library in htmlwidgets/lib,
<http://d3js.org>),
David Chazalviel [aut, cre],
Harry Stevens [cph] (d3-regression library in htmlwidgets/lib,
<https://github.com/HarryStevens/d3-regression>)

Maintainer David Chazalviel <david.chazalviel@club-internet.fr>

Repository CRAN

Date/Publication 2021-10-21 17:00:02 UTC

R topics documented:

<code>changeMouseMode</code>	2
<code>getPlotConfig</code>	3
<code>scatterPlotMatrix</code>	3
<code>scatterPlotMatrix-shiny</code>	6

setCategoricalColorScale	6
setContinuousColorScale	7
setCorrPlotCS	9
setCorrPlotType	10
setDistribType	11
setKeptColumns	12
setRegressionType	13
setZAxis	14

Index	16
--------------	-----------

changeMouseMove	<i>This function allows to set the type of interaction; three types of mouse interactions are available ('tooltip', 'filter' or 'zoom').</i>
-----------------	--

Description

This function allows to set the type of interaction; three types of mouse interactions are available ('tooltip', 'filter' or 'zoom').

Usage

```
changeMouseMove(id, interactionType)
```

Arguments

id	Output variable to read from (id which references the requested plot).
interactionType	Type of mouse interaction.

Value

No return value, called from shiny applications for side effects.

Examples

```
if(interactive()) {
  library(shiny)
  library(scatterPlotMatrix)

  ui <- fluidPage(
    selectInput(
      "mouseMode",
      "Mouse Interactions:",
      c("Tooltip" = "tooltip", "Filter" = "filter", "Zoom" = "zoom")
    ),
    p("The selector controls the type of mouse interactions with the scatterPlotMatrix"),
    scatterPlotMatrixOutput("spMatrix")
  )
}
```

```
server <- function(input, output, session) {  
  output$spMatrix <- renderScatterPlotMatrix({  
    scatterPlotMatrix(iris)  
  })  
  observe({  
    scatterPlotMatrix::changeMouseMode("spMatrix", input$mouseMode)  
  })  
}  
  
shinyApp(ui, server)  
}
```

getPlotConfig	<i>Asks to retrieve the plot configuration. Result will be sent through a reactive input.</i>
---------------	---

Description

Asks to retrieve the plot configuration. Result will be sent through a reactive input.

Usage

```
getPlotConfig(id, configInputId)
```

Arguments

`id` Output variable to read from (id which references the requested plot).
`configInputId` Reactive input to write to.

Value

No return value, called from shiny applications for side effects.

scatterPlotMatrix	<i>htmlwidget for d3.js scatter plot matrix</i>
-------------------	---

Description

htmlwidget for d3.js scatter plot matrix

Usage

```
scatterPlotMatrix(
  data,
  controlWidgets = FALSE,
  categorical = NULL,
  inputColumns = NULL,
  keptColumns = NULL,
  zAxisDim = NULL,
  distribType = 2,
  regressionType = 0,
  corrPlotType = "Circles",
  corrPlotCS = "Plasma",
  rotateTitle = FALSE,
  columnLabels = NULL,
  continuousCS = "Viridis",
  categoricalCS = "Category10",
  eventInputId = NULL,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

<code>data</code>	data.frame with data to use in the chart.
<code>controlWidgets</code>	Tells if some widgets must be available to control plot; NULL is allowed, meaning that <code>'!HTMLWidgets.shinyMode'</code> is to use; default value is FALSE.
<code>categorical</code>	List of list (one for each data column) containing the name of available categories, or NULL if column corresponds to continuous data; NULL is allowed, meaning all columns are continuous.
<code>inputColumns</code>	List of boolean (one for each data column), TRUE for an input column, FALSE for an output column; NULL is allowed, meaning all columns are inputs.
<code>keptColumns</code>	List of boolean (one for each data column), FALSE if column has to be ignored; NULL is allowed, meaning all columns are available.
<code>zAxisDim</code>	Name of the column represented by z axis (used to determine the color to attribute to a point); NULL is allowed, meaning there is no coloring to apply.
<code>distribType</code>	Binary code indicating the type of distribution plot (bit 1: density plot, bit 2: histogram).
<code>regressionType</code>	Binary code indicating the type of regression plot (bit 1: linear, bit 2: loess).
<code>corrPlotType</code>	String indicating the type of correlation plots to use; supported values: <code>Circles</code> to use a circle tree map, <code>Text</code> to use simple text labels; default value is <code>Circles</code> .
<code>corrPlotCS</code>	Name of the color Scale to use for correlation plot when plot type is <code>'Text'</code> ; supported names: <code>"Viridis"</code> , <code>"Inferno"</code> , <code>"Magma"</code> , <code>"Plasma"</code> , <code>"Warm"</code> , <code>"Cool"</code> , <code>"Rainbow"</code> , <code>"CubehelixDefault"</code> , <code>"Blues"</code> , <code>"Greens"</code> , <code>"Greys"</code> , <code>"Oranges"</code> , <code>"Purples"</code> , <code>"Reds"</code> , <code>"BuGn"</code> , <code>"BuPu"</code> , <code>"GnBu"</code> , <code>"OrRd"</code> , <code>"PuBuGn"</code> , <code>"PuBu"</code> , <code>"PuRd"</code> ,

	"RdBu", "RdPu", "YlGnBu", "YlGn", "YlOrBr", "YlOrRd"; default value is Plasma.
rotateTitle	TRUE if column title must be rotated.
columnLabels	List of string (one for each data column) to display in place of column name found in data, or NULL if there is no alternative name; NULL is allowed, meaning all columns are without alternative name; can be used to insert line breaks.
continuousCS	Name of the color Scale to use for continuous data; supported names: "Viridis", "Inferno", "Magma", "Plasma", "Warm", "Cool", "Rainbow", "CubehelixDefault", "Blues", "Greens", "Greys", "Oranges", "Purples", "Reds", "BuGn", "BuPu", "GnBu", "OrRd", "PuBuGn", "PuBu", "PuRd", "RdBu", "RdPu", "YlGnBu", "YlGn", "YlOrBr", "YlOrRd"; default value is Viridis.
categoricalCS	Name of the color Scale to use for categorical data; supported names: Category10, Accent, Dark2, Paired, Set1; default value is Category10.
eventInputId	When plot event occurred, reactive input to write to; NULL is allowed, default value is 'plotEvent'.
width	Integer in pixels defining the width of the widget.
height	Integer in pixels defining the height of the widget.
elementId	Unique CSS selector id for the widget.

Examples

```

if(interactive()) {
  library(scatterPlotMatrix)

  scatterPlotMatrix(iris, zAxisDim = "Species")
  # Each point has a color depending of its 'Species' value

  categorical <- list(NULL, c(4, 6, 8), NULL, NULL, NULL, NULL, NULL, c(0, 1), c(0, 1), 3:5, 1:8)
  scatterPlotMatrix(mtcars, categorical = categorical, zAxisDim = "cyl")
  # 'cyl' and four last columns have a box representation for its categories
  # (use top slider to see the last three columns)

  scatterPlotMatrix(iris, zAxisDim = "Species", distribType = 1)
  # Distribution plots are of type 'density plot' (instead of histogram)

  scatterPlotMatrix(iris, zAxisDim = "Species", regressionType = 1)
  # Add linear regression plots

  columnLabels <- gsub("\\.", "<br>", colnames(iris))
  scatterPlotMatrix(iris, zAxisDim = "Species", columnLabels = columnLabels)
  # Given names are displayed in place of dataset column names; <br> is used to insert line breaks
}

```

 scatterPlotMatrix-shiny

Shiny bindings for scatterPlotMatrix

Description

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

Usage

```
scatterPlotMatrixOutput(outputId, width = "100%", height = "600px")
```

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

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a scatterPlotMatrix
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.

 setCategoricalColorScale

Tells which color scale to use for categorical columns.

Description

Tells which color scale to use for categorical columns.

Usage

```
setCategoricalColorScale(id, categoricalCsId)
```

Arguments

id	Output variable to read from (id which references the requested plot).
categoricalCsId	One of the available color scale ids (Category10, Accent, Dark2, Paired, Set1).

Value

No return value, called from shiny applications for side effects.

Examples

```
if(interactive()) {
  library(shiny)
  library(scatterPlotMatrix)

  ui <- fluidPage(
    selectInput(
      "categoricalCsSelect",
      "Categorical Color Scale:",
      choices = list(
        "Category10" = "Category10", "Accent" = "Accent", "Dark2" = "Dark2",
        "Paired" = "Paired", "Set1" = "Set1"
      ),
      selected = "Category10"
    ),
    p("The selector controls the colors used when reference column is of type categorical"),
    scatterPlotMatrixOutput("spMatrix")
  )

  server <- function(input, output, session) {
    output$spMatrix <- renderScatterPlotMatrix({
      scatterPlotMatrix(iris, zAxisDim = "Species")
    })
    observeEvent(input$categoricalCsSelect, {
      scatterPlotMatrix::setCategoricalColorScale("spMatrix", input$categoricalCsSelect)
    })
  }

  shinyApp(ui, server)
}
```

setContinuousColorScale

Tells which color scale to use for continuous columns.

Description

Tells which color scale to use for continuous columns.

Usage

```
setContinuousColorScale(id, continuousCsId)
```

Arguments

`id` Output variable to read from (id which references the requested plot).

`continuousCsId` One of the available color scale ids ("Viridis", "Inferno", "Magma", "Plasma", "Warm", "Cool", "Rainbow", "CubehelixDefault", "Blues", "Greens", "Greys", "Oranges", "Purples", "Reds", "BuGn", "BuPu", "GnBu", "OrRd", "PuBuGn", "PuBu", "PuRd", "RdBu", "RdPu", "YlGnBu", "YlGn", "YlOrBr", "YlOrRd").

Value

No return value, called from shiny applications for side effects.

Examples

```
if(interactive()) {
  library(shiny)
  library(scatterPlotMatrix)

  ui <- fluidPage(
    selectInput(
      "continuousCsSelect",
      "Continuous Color Scale:",
      choices = list(
        "Viridis" = "Viridis", "Inferno" = "Inferno", "Magma" = "Magma",
        "Plasma" = "Plasma", "Warm" = "Warm", "Cool" = "Cool", "Rainbow" = "Rainbow",
        "CubehelixDefault" = "CubehelixDefault", "Blues" = "Blues",
        "Greens" = "Greens", "Greys" = "Greys", "Oranges" = "Oranges",
        "Purples" = "Purples", "Reds" = "Reds", "BuGn" = "BuGn", "BuPu" = "BuPu",
        "GnBu" = "GnBu", "OrRd" = "OrRd", "PuBuGn" = "PuBuGn", "PuBu" = "PuBu",
        "PuRd" = "PuRd", "RdBu" = "RdBu", "RdPu" = "RdPu", "YlGnBu" = "YlGnBu",
        "YlGn" = "YlGn", "YlOrBr" = "YlOrBr", "YlOrRd" = "YlOrRd"
      ),
      selected = "Viridis"
    ),
    p("The selector controls the colors used when reference column is of type continuous"),
    scatterPlotMatrixOutput("spMatrix")
  )

  server <- function(input, output, session) {
    output$spMatrix <- renderScatterPlotMatrix({
      scatterPlotMatrix(iris, zAxisDim = "Sepal.Length")
    })
    observeEvent(input$continuousCsSelect, {
      scatterPlotMatrix::setContinuousColorScale("spMatrix", input$continuousCsSelect)
    })
  }

  shinyApp(ui, server)
}
```

setCorrPlotCS	<i>Tells which color scale to use for correlation plots.</i>
---------------	--

Description

Tells which color scale to use for correlation plots.

Usage

```
setCorrPlotCS(id, corrPlotCsId)
```

Arguments

id	Output variable to read from (id which references the requested plot).
corrPlotCsId	One of the available color scale ids ("Viridis", "Inferno", "Magma", "Plasma", "Warm", "Cool", "Rainbow", "CubehelixDefault", "Blues", "Greens", "Greys", "Oranges", "Purples", "Reds", "BuGn", "BuPu", "GnBu", "OrRd", "PuBuGn", "PuBu", "PuRd", "RdBu", "RdPu", "YlGnBu", "YlGn", "YlOrBr", "YlOrRd").

Value

No return value, called from shiny applications for side effects.

Examples

```
if(interactive()) {
  library(shiny)
  library(scatterPlotMatrix)

  ui <- fluidPage(
    selectInput(
      "corrPlotCsSelect",
      "Correlation Plot Color Scale:",
      choices = list(
        "Viridis" = "Viridis", "Inferno" = "Inferno", "Magma" = "Magma",
        "Plasma" = "Plasma", "Warm" = "Warm", "Cool" = "Cool", "Rainbow" = "Rainbow",
        "CubehelixDefault" = "CubehelixDefault", "Blues" = "Blues",
        "Greens" = "Greens", "Greys" = "Greys", "Oranges" = "Oranges",
        "Purples" = "Purples", "Reds" = "Reds", "BuGn" = "BuGn", "BuPu" = "BuPu",
        "GnBu" = "GnBu", "OrRd" = "OrRd", "PuBuGn" = "PuBuGn", "PuBu" = "PuBu",
        "PuRd" = "PuRd", "RdBu" = "RdBu", "RdPu" = "RdPu", "YlGnBu" = "YlGnBu",
        "YlGn" = "YlGn", "YlOrBr" = "YlOrBr", "YlOrRd" = "YlOrRd"
      ),
      selected = "Plasma"
    ),
    p("The selector controls the color scale to use for correlation plot
      when plot type is 'Text'"),
    scatterPlotMatrixOutput("spMatrix")
  )
}
```

```

server <- function(input, output, session) {
  output$spMatrix <- renderScatterPlotMatrix({
    scatterPlotMatrix(iris, corrPlotType = "Text")
  })
  observeEvent(input$corrPlotCsSelect, {
    scatterPlotMatrix::setCorrPlotCS("spMatrix", input$corrPlotCsSelect)
  })
}

shinyApp(ui, server)
}

```

setCorrPlotType	<i>Tells which type of correlation plot to use.</i>
-----------------	---

Description

Tells which type of correlation plot to use.

Usage

```
setCorrPlotType(id, corrPlotType)
```

Arguments

`id` Output variable to read from (id which references the requested plot).
`corrPlotType` One of the available correlation plot types (Circles, Text).

Value

No return value, called from shiny applications for side effects.

Examples

```

if(interactive()) {
  library(shiny)
  library(scatterPlotMatrix)

  ui <- fluidPage(
    selectInput(
      "corrPlotTypeSelect",
      "Correlation Plot Type:",
      choices = list("Circles" = "Circles", "Text" = "Text"),
      selected = "Circles"
    ),
    p("The selector controls the type of correlation to use"),
    scatterPlotMatrixOutput("spMatrix")
  )
}

```

```

)

server <- function(input, output, session) {
  output$spMatrix <- renderScatterPlotMatrix({
    scatterPlotMatrix(iris, zAxisDim = "Sepal.Length", continuousCS = "Plasma")
  })
  observeEvent(input$corrPlotTypeSelect, {
    scatterPlotMatrix::setCorrPlotType("spMatrix", input$corrPlotTypeSelect)
  })
}

shinyApp(ui, server)
}

```

setDistribType	<i>Tells which type of representation to use for distribution plots.</i>
----------------	--

Description

Tells which type of representation to use for distribution plots.

Usage

```
setDistribType(id, distribType)
```

Arguments

id	Output variable to read from (id which references the requested plot).
distribType	Binary code indicating the type of distribution plot (bit 1: histogram, bit 2: density plot).

Value

No return value, called from shiny applications for side effects.

Examples

```

if(interactive()) {
  library(shiny)
  library(scatterPlotMatrix)

  ui <- fluidPage(
    selectInput(
      "distribType",
      "Distribution Representation:",
      choices = list("Histogram" = 2, "Density Plot" = 1),
      selected = 2
    ),

```

```

    p("The selector controls type of representation to use for distribution plots"),
    scatterPlotMatrixOutput("spMatrix")
  )

  server <- function(input, output, session) {
    output$spMatrix <- renderScatterPlotMatrix({
      scatterPlotMatrix(iris)
    })
    observeEvent(input$distribType, {
      scatterPlotMatrix::setDistribType("spMatrix", input$distribType)
    })
  }

  shinyApp(ui, server)
}

```

 setKeptColumns

Column visibility

Description

Tells which columns have to be visible.

Usage

```
setKeptColumns(id, keptColumns)
```

Arguments

id	Output variable to read from (id which references the requested plot).
keptColumns	Vector of boolean (one for each data column), FALSE if column has to be hidden. A named list can also be provided to only indicate which columns must be assigned to a new visibility.

Value

No return value, called from shiny applications for side effects.

Examples

```

if(interactive()) {
  library(shiny)
  library(scatterPlotMatrix)

  ui <- fluidPage(
    checkboxInput("hideColumnsCB", "Hide last columns", FALSE),
    p("The check box controls the visibility of the two last columns"),
    scatterPlotMatrixOutput("spMatrix")
  )
}

```

```

)

server <- function(input, output, session) {
  output$spMatrix <- renderScatterPlotMatrix({
    scatterPlotMatrix(iris)
  })
  observeEvent(input$hideColumnsCB, {
    keptColumns <- vapply(
      1:ncol(iris),
      function(i) {
        return(iffelse(input$hideColumnsCB, ncol(iris) - i >= 2, TRUE))
      },
      logical(1)
    )
    scatterPlotMatrix::setKeptColumns("spMatrix", keptColumns)
  })
}

shinyApp(ui, server)
}

```

setRegressionType *Tells which type of regression to use for regression plots.*

Description

Tells which type of regression to use for regression plots.

Usage

```
setRegressionType(id, regressionType)
```

Arguments

`id` Output variable to read from (id which references the requested plot).
`regressionType` Binary code indicating the type of regression plot (bit 1: linear, bit 2: loess).

Value

No return value, called from shiny applications for side effects.

Examples

```

if(interactive()) {
  library(shiny)
  library(scatterPlotMatrix)

  ui <- fluidPage(

```

```

checkboxInput("linearRegressionCB", "Linear Regression", FALSE),
checkboxInput("loessCB", "Local Polynomial Regression", FALSE),
p("The check boxes controls type of regression to use for regression plots"),
scatterPlotMatrixOutput("spMatrix")
)

server <- function(input, output, session) {
  output$spMatrix <- renderScatterPlotMatrix({
    scatterPlotMatrix(iris)
  })
  observe({
    linearFlag <- ifelse(input$linearRegressionCB, 1, 0)
    loessFlag <- ifelse(input$loessCB, 2, 0)
    scatterPlotMatrix::setRegressionType("spMatrix", linearFlag + loessFlag)
  })
}

shinyApp(ui, server)
}

```

setZAxis

Tells which dim is to display on Z axis.

Description

Tells which dim is to display on Z axis.

Usage

```
setZAxis(id, dim)
```

Arguments

id	Output variable to read from (id which references the requested plot).
dim	Is to display on X axis.

Value

No return value, called from shiny applications for side effects.

Examples

```

if(interactive()) {
  library(shiny)
  library(scatterPlotMatrix)

  ui <- fluidPage(
    fluidRow(

```

```
      column(
        2,
        selectInput("zAxisSelect", "Z Axis:", colnames(iris))
      ),
      column(
        2,
        checkboxInput("zAxisUsedCB", "Use Z Axis", FALSE)
      )
    ),
    scatterPlotMatrixOutput("spMatrix")
  )

server <- function(input, output, session) {
  output$spMatrix <- renderScatterPlotMatrix({
    scatterPlotMatrix(iris)
  })

  observe({
    scatterPlotMatrix::setZAxis("spMatrix", if (input$zAxisUsedCB) input$zAxisSelect else NULL)
  })
}

shinyApp(ui, server)
}
```

Index

changeMouseMode, [2](#)
getPlotConfig, [3](#)
renderScatterPlotMatrix
 (scatterPlotMatrix-shiny), [6](#)
scatterPlotMatrix, [3](#)
scatterPlotMatrix-shiny, [6](#)
scatterPlotMatrixOutput
 (scatterPlotMatrix-shiny), [6](#)
setCategoricalColorScale, [6](#)
setContinuousColorScale, [7](#)
setCorrPlotCS, [9](#)
setCorrPlotType, [10](#)
setDistribType, [11](#)
setKeptColumns, [12](#)
setRegressionType, [13](#)
setZAxis, [14](#)