Package 'parallelPlot'

January 18, 2021

Title 'Htmlwidget' for a Parallel Coordinates Plot

Version 0.1.0

Description Create a parallel coordinates plot, using 'htmlwidgets' package and 'd3.js'.

URL https://gitlab.com/drti/parallelplot

BugReports https://gitlab.com/drti/parallelplot/-/issues

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

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],
Benoit Lehman [aut]

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

Repository CRAN

Date/Publication 2021-01-18 09:10:03 UTC

R topics documented:

---

changeRow *Row edition*

Description

Asks to change a row.

Usage

```
changeRow(id, rowIndex, newValues)
```

Arguments

id	output variable to read from (id which references the requested plot)
rowIndex	index of the changed row.
newValues	list of new values to attribute to the row (list associating a value to a column identifier).

Value

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

Examples

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

  ui <- fluidPage(
      sliderInput("rowValueSlider", "Value for 'Sepal.Length' of first row:",
          min = 4, max = 8, step = 0.1, value = iris[["Sepal.Length"]][1]),
    p("The slider controls the new value to assign to the 'Sepal.Length' of the first row"),
      parallelPlotOutput("parPlot")
  )

  server <- function(input, output, session) {
      output$parPlot <- renderParallelPlot({
          parallelPlot(iris)
      })
      observeEvent(input$rowValueSlider, {
          newValues <- iris[1,]
          newValues[["Sepal.Length"]] <- input$rowValueSlider
```

```
            parallelPlot::changeRow("parPlot", 1, newValues)
        })
    }

    shinyApp(ui, server)
}
```

---

getValue *Plot attributes*

Description

Asks to retrieve the value of an attribute.

Usage

```
getValue(id, attrType, valueInputId)
```

Arguments

id	output variable to read from (id which references the requested plot)
attrType	which value is requested.
valueInputId	reactive input to write to.

Details

Available attributes are 'Cutoffs', 'SelectedTraces' and 'ReferenceColumn'. Result will be sent through a reactive input.

Value

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

Examples

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

  ui <- fluidPage(
      actionButton("getSelectedTracesAction", "Retrieve Selected Traces"),
      p("The button displays the list of uncutted rows (use brush to reduce it)"),
      parallelPlotOutput("parPlot")
  )

  server <- function(input, output, session) {
      output$parPlot <- renderParallelPlot({
          parallelPlot(iris)
```

```
        })
        observeEvent(input$getSelectedTracesAction, {
            attributeType <- "SelectedTraces"
            parallelPlot::getValue("parPlot", attributeType, "MySelectedTraces")
        })
        observeEvent(input$MySelectedTraces, {
            showModal(modalDialog(
                title = "Selected Traces",
                toString(input$MySelectedTraces)
            ))
        })
    }

    shinyApp(ui, server)
}
```

`parallelPlot` *htmlwidget for d3.js parallel coordinate plot*

Description

htmlwidget for d3.js parallel coordinate plot

Usage

```
parallelPlot(
  data,
  categorical = NULL,
  inputColumns = NULL,
  keptColumns = NULL,
  histoVisibility = NULL,
  cutoffs = NULL,
  refRowIndex = NULL,
  refColumnDim = NULL,
  rotateTitle = FALSE,
  columnLabels = NULL,
  continuousCS = "Blues",
  categoricalCS = "Category10",
  eventInputId = NULL,
  editionMode = "EditionOff",
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

data	data.frame with data to use in the chart.
categorical	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.
inputColumns	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.
keptColumns	List of boolean (one for each data column), FALSE if column has to be ignored; NULL is allowed, meaning all columns are available.
histoVisibility	List of boolean (one for each data column), TRUE if an histogram must be displayed; NULL is allowed, meaning no histogram must be displayed.
cutoffs	List of list (one for each data column) of list (one for each cutoff) containing two values (min and max values defining the cutoff) or NULL if there is no cutoff to apply; NULL is allowed, meaning all columns are without cutoff.
refRowIndex	Index of the sample row which has to appear horizontal; NULL is allowed, meaning there is no row to use as reference.
refColumnDim	Name of the reference column (used to determine the color to attribute to a row); NULL is allowed, meaning there is no coloring to apply.
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: Blues,RdBu,YlGnBu,YlOrRd,Reds default value is Blues).
categoricalCS	Name of the color Scale to use for categorical data (supported names: Category10,Accent,Dark2,Paire default value is Category10).
eventInputId	When plot event occurred, reactive input to write to; NULL is allowed, default value is 'plotEvent'.
editionMode	Supported edition modes: EditionOff,EditionOnDrag,EditionOnDragEnd; default value is EditionOff .
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.

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

Examples

```
if(interactive()) {
  library(parallelPlot)

 categorical <- list(NULL, c(4, 6, 8), NULL, NULL, NULL, NULL, NULL, c(0, 1), c(0, 1), 3:5, 1:8)
  parallelPlot(mtcars, categorical = categorical, refColumnDim = "cyl")
  # 'cyl' and four last columns have a box representation for its categories

  histoVisibility <- rep(TRUE, ncol(iris))
  parallelPlot(iris, histoVisibility = histoVisibility)
  # An histogram is displayed for each column

  histoVisibility <- rep(TRUE, ncol(iris))
  cutoffs <- list(list(c(6, 7)), NULL, NULL, NULL, c("virginica", "setosa"))
  parallelPlot(iris, histoVisibility = histoVisibility, cutoffs = cutoffs)
  # Cut traces are greyed; an histogram is displayed considering only kept traces

  parallelPlot(iris, refRowIndex = 1)
 # Axes are shifted vertically in such a way that first trace of the dataset looks horizontal

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

`parallelPlot-shiny` *Shiny bindings for parallelPlot*

Description

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

Usage

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

renderParallelPlot(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 parallelPlot
`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.

Value

An output or render function that enables the use of the widget within Shiny applications.

`setCategoricalColorScale` *Traces colors*

Description

Tells which color scale to use when reference column is of type categorical.

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

Details

If a column is defined as the reference (for example by clicking on its header), a color scale is associated to this column. Available color scale ids are: 'Category10', 'Accent', 'Dark2', 'Paired', 'Set1'.

Value

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

Examples

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

  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"),
      parallelPlotOutput("parPlot")
  )

  server <- function(input, output, session) {
      output$parPlot <- renderParallelPlot({
          parallelPlot(data = iris, refColumnDim = "Species")
```

```
    })
    observeEvent(input$categoricalCsSelect, {
        parallelPlot::setCategoricalColorScale("parPlot", input$categoricalCsSelect)
    })
  }

  shinyApp(ui, server)
}
```

---

setContinuousColorScale *Traces colors*

Description

Tells which color scale to use when reference column is of type continuous.

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

Details

If a column is defined as the reference (for example by clicking on its header), a color scale is associated to this column. Available color scale ids are: 'Blues', 'RdBu', 'YlGnBu', 'YlOrRd', 'Reds'.

Value

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

Examples

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

  ui <- fluidPage(
      selectInput("continuousCsSelect", "Continuous Color Scale:",
          choices = list("Blues" = "Blues", "RdBu" = "RdBu", "YlGnBu" = "YlGnBu",
                         "YlOrRd" = "YlOrRd", "Reds" = "Reds"), selected = "Blues"),
    p("The selector controls the colors used when reference column is of type continuous"),
      parallelPlotOutput("parPlot")
```

```
  )

  server <- function(input, output, session) {
      output$parPlot <- renderParallelPlot({
          parallelPlot(iris, refColumnDim = "Sepal.Length")
      })
      observeEvent(input$continuousCsSelect, {
          parallelPlot::setContinuousColorScale("parPlot", input$continuousCsSelect)
      })
  }

  shinyApp(ui, server)
}
```

`setCutoffs` *Cutoffs values*

Description

Tells which cutoffs to use for each column.

Usage

```
setCutoffs(id, cutoffs)
```

Arguments

`id`	output variable to read from (id which references the requested plot)
`cutoffs`	Vector of list (one for each data column) of vector (one for each cutoff) containing two values for continuous input (min and max value defining the cutoff), or one value for categorical input (name of the category to keep), or `NULL` if there is no cutoff to apply; `NULL` is allowed, meaning all columns are without cutoff. A named list can also be provided to only indicate which columns must be assigned to a new cutoff.

Details

It's possible to filter some traces by defining cutoffs to apply to columns.

Value

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

Examples

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

  ui <- fluidPage(
      sliderInput("brushSlider", "Brush for 'Sepal.Length' column:",
          min = 4, max = 8, step = 0.1, value = c(4, 8)),
      p("The slider controls the rows which are kept by cutoff (others are greyed)"),
      parallelPlotOutput("parPlot")
  )

  server <- function(input, output, session) {
      output$parPlot <- renderParallelPlot({
          parallelPlot(iris)
      })
      observeEvent(input$brushSlider, {
          cutoffs <- list()
          cutoffs["Sepal.Length"] <- list(list(input$brushSlider))
          parallelPlot::setCutoffs("parPlot", cutoffs)
      })
  }

  shinyApp(ui, server)
}
```

`setHistoVisibility` *Histograms visibility*

Description

Tells which columns have to be displayed with histograms.

Usage

```
setHistoVisibility(id, histoVisibility)
```

Arguments

`id`	output variable to read from (id which references the requested plot)
`histoVisibility`	Vector of boolean (one for each data column), `TRUE` if an histogram must be displayed; `NULL` is allowed, meaning no histogram must be displayed. A named list can also be provided to only indicate which columns must be assigned to a new display.

Value

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

Examples

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

  ui <- fluidPage(
      checkboxInput("histCB", "Histogram Visibility", FALSE),
      p("The check box controls the visibility of histograms"),
      parallelPlotOutput("parPlot")
  )

  server <- function(input, output, session) {
      output$parPlot <- renderParallelPlot({
          parallelPlot(iris)
      })
      observeEvent(input$histCB, {
          histoVisibility <- rep(input$histCB, ncol(iris))
          parallelPlot::setHistoVisibility("parPlot", histoVisibility)
      })
  }

  shinyApp(ui, server)
}
```

`setKeptColumns` *Column visibitity*

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(parallelPlot)

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

  server <- function(input, output, session) {
      output$parPlot <- renderParallelPlot({
          parallelPlot(mtcars)
      })
      observeEvent(input$hideColumnsCB, {
          keptColumns <- sapply(1:ncol(mtcars), function(i) {
              return(ifelse(input$hideColumnsCB, ncol(mtcars) - i >= 2, TRUE))
          })
          parallelPlot::setKeptColumns("parPlot", keptColumns)
      })
  }

  shinyApp(ui, server)
}
```

Index