

Package ‘parcoords’

May 24, 2019

Title 'Htmlwidget' for 'd3.js' Parallel Coordinates Chart

Version 1.0.0

Date 2019-05-13

Maintainer Kenton Russell <kent.russell@timelyportfolio.com>

Description Create interactive parallel coordinates charts with this 'htmlwidget' wrapper for 'd3.js' <<https://github.com/BigFatDog/parcoords-es>> {'parallel-coordinates'}.

URL <https://github.com/timelyportfolio/parcoords>

BugReports <https://github.com/timelyportfolio/parcoords/issues>

Depends R (>= 3.2.0)

License MIT + file LICENSE

LazyData true

Imports crosstalk, htmlwidgets (>= 0.6.0), utils

Suggests d3r, ggplot2, htmltools, knitr, shiny, testthat, rmarkdown

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Mike Bostock [aut, cph] (d3.js library in htmlwidgets/lib, <http://d3js.org>),
Kai Chang [aut, cph] (parallel coordinates reusable chart, <https://github.com/syntagmatic/parallel-coordinates>),
Xing Yun [aut, cph] ('es6' parallel coordinates, <https://github.com/BigFatDog/parcoords-es>),
Kenton Russell [aut, cre] (R interface),
Anobel Odisho [aut] (R interface guidance, suggestions, testing, review, and feedback),
Mark Albrecht [ctb] (Shiny output and example)

Repository CRAN

Date/Publication 2019-05-24 13:10:03 UTC

R topics documented:

parcoords	2
parcoords-shiny	5
parcoordsProxy	8
pcCenter	9
pcFilter	10
pcHide	10
pcSnapshot	11
pcUnhide	11

Index	12
--------------	-----------

parcoords	<i>Interactive 'd3.js' Parallel Coordinates Chart</i>
-----------	---

Description

Create interactive parallel coordinates charts with this htmlwidget wrapper for d3.js [parallel-coordinates](#).

Usage

```
parcoords(data = NULL, rownames = TRUE, color = NULL,
  brushMode = NULL, brushPredicate = "and", alphaOnBrushed = NULL,
  reorderable = FALSE, axisDots = NULL, margin = NULL,
  composite = NULL, alpha = NULL, queue = FALSE, mode = FALSE,
  rate = NULL, dimensions = NULL, bundleDimension = NULL,
  bundlingStrength = 0.5, smoothness = 0, tasks = NULL,
  autoresize = FALSE, withD3 = FALSE, width = NULL, height = NULL,
  elementId = NULL)
```

Arguments

data	data.frame with data to use in the chart
rownames	logical use rownames from the data.frame in the chart. Regardless of this parameter, we will append rownames to the data that we send to JavaScript. If rownames equals FALSE, then we will use parallel coordinates to hide it.
color	Color can be a single color as rgb or hex. For a color function, provide a list(colorScale = , colorBy = , colorScheme = , colorInterpolator = , colorDomain =) where colorScale is the name of the d3-scale such as scaleOrdinal or scaleSequential, colorBy with the column name from the data to determine color. If applying color to a discrete or ordinal variable then please also supply colorScheme, such as schemCategory10. If applying color to a continuous variable then please also supply colorInterpolator with colorInterpolator as the name of the d3 interpolator , such as interpolateViridis. If using a d3 color scale, then make sure to use the argument withD3 = TRUE.
brushMode	string, either "1D-axes", "1D-axes-multi", or "2D-strums" giving the type of desired brush behavior for the chart.

brushPredicate	string, either "and" or "or" giving the logic for the join with multiple brushes.
alphaOnBrushed	opacity from 0 to 1 when brushed (default to 0).
reorderable	logical enable reordering of axes
axisDots	logical mark the points where polylines meet an axis with dots
margin	list of sizes of margins in pixels. Currently brushMode = "2D-strums" requires left margin = 0, so this will change automatically and might result in unexpected behavior.
composite	foreground context's composite type
alpha	opacity from 0 to 1 of the polylines
queue	logical (default FALSE) to change rendering mode to queue for progressive rendering. Usually queue = T for very large datasets.
mode	string see queue above; queue = T will set mode = "queue"
rate	integer rate at which render will queue
dimensions	list to customize axes dimensions
bundleDimension	character string for the column or variable on which to bundle
bundlingStrength	numeric value between 0 and 1 for the strength of the bundling. This value will not affect the parallel coordinates if bundleDimension is not set and will be ignored.
smoothness	numeric value between 0 and 1 for strength of smoothing or curvature. This value will not affect the parallel coordinates if bundleDimension is not set and will be ignored.
tasks	a character string or JS or list of strings or JS representing a JavaScript function(s) to run after the parcoords has rendered. These provide an opportunity for advanced customization. Note, the function will use the JavaScript call mechanism, so within the function, this will be an object with this.el representing the containing element of the parcoords and this.parcoords representing the parcoords instance.
autoresize	logical (default FALSE) to auto resize the parcoords when the size of the container changes. This is useful in contexts such as rmarkdown slide presentations or flexdashboard. However, this will not be useful if you expect bigger data or a more typical html context.
withD3	logical to include d3 dependency from d3r. The 'parcoords' htmlwidget uses a standalone JavaScript build and will not include the entire d3 in the global/window namespace. To include d3.js in this way, use withD3=TRUE.
width	integer in pixels defining the width of the widget. Autosizing to 100 of the widget container will occur if width = NULL .
height	integer in pixels defining the height of the widget. Autosizing to 400px of the widget container will occur if height = NULL .
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()) {
  # simple example using the mtcars dataset
  data( mtcars )
  parcoords( mtcars )

  # various ways to change color
  # in these all lines are the specified color
  parcoords( mtcars, color = "green" )
  parcoords( mtcars, color = "#f0c" )
  # in these we supply a function for our color
  parcoords(
    mtcars
    , color = list(
      colorBy = "cyl"
      , colorScale = "scaleOrdinal"
      , colorScheme = "schemeCategory10"
    )
    , withD3 = TRUE
  )

  if(require('ggplot2', quietly = TRUE)) {
    parcoords(
      diamonds
      ,rownames = FALSE
      ,brushMode = "1d-axes"
      ,reorderable = TRUE
      ,queue = TRUE
      ,color= list(
        colorBy="cut"
        , colorScale = "scaleOrdinal"
        , colorScheme = "schemeCategory10"
      )
      ,withD3 = TRUE
    )
  }
}
library(parcoords)

parcoords(
  mtcars,
  dimensions = list(
    cyl = list(
      title = "cylinder",
      tickValues = unique(mtcars$cyl)
    )
  )
)

```

```

)

parcoords(
  mtcars
  ,rownames = FALSE
  ,brushMode = "1d-multi"
  ,brushPredicate = "OR"
  ,dimensions = list(
    cyl = list(
      title = "cylinder",
      tickValues = unique(mtcars$cyl)
    )
  )
)
)

```

parcoords-shiny

Shiny bindings for 'parcoords'

Description

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

Usage

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

```
renderParcoords(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 sunburst
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()) {
  #### filter proxy example ----

  library(parcoords)
  library(shiny)

  ui <- tagList(

```

```

    textOutput("filteredstate", container=h3),
    parcoordsOutput("pc")
  )

server <- function(input, output, session) {
  rv <- reactiveValues(filtered = FALSE)

  output$pc <- renderParcoords({
    parcoords(mtcars)
  })

  observe({
    # toggle between filtered and unfiltered every 2.5 seconds
    invalidateLater(2500)
    rv$filtered <- !isolate(rv$filtered)
  })

  observeEvent(rv$filtered, {
    # create a proxy with which we will communicate between
    # Shiny and the parallel coordinates without a re-render
    pcp <- parcoordsProxy("pc")

    if(rv$filtered) {
      pcFilter(
        pcp,
        list(
          cyl = c(6,8),
          hp = list(gt = 200)
        )
      )
    } else {
      pcFilter(pcp, list())
    }
  })

  output$filteredstate <- renderText({
    paste0("Filtered: ", rv$filtered)
  })
}

shinyApp(ui = ui, server = server)

### center proxy example ----
library(shiny)
library(parcoords)

ui <- tags$div(
  parcoordsOutput("pc", width = 2500),
  style="width: 2500px;"
)

server <- function(input, output, session) {
  # create a proxy with which we will communicate between

```

```

# Shiny and the parallel coordinates without a re-render
pcp <- parcoordsProxy("pc")

output$pc <- renderParcoords({
  parcoords(mtcars)
})

pcCenter(pcp, 'drat')
}

shinyApp(ui=ui, server=server)

### hide/unhide proxy example ----
library(parcoords)
library(shiny)

ui <- tagList(
  selectizeInput(
    inputId = "columns",
    label = "Columns to Hide",
    choices = c("names", colnames(mtcars)),
    selected = "names",
    multiple = TRUE
  ),
  parcoordsOutput("pc"),
  checkboxInput("hidenames", label="Hide Row Names", value=TRUE),
  parcoordsOutput("pc2")
)

server <- function(input, output, session) {
  output$pc <- renderParcoords({
    parcoords(mtcars, rownames = FALSE, brushMode = "1d")
  })

  output$pc2 <- renderParcoords({
    parcoords(mtcars, rownames = FALSE)
  })

  pcUnhide

  observeEvent(input$columns, {
    # create a proxy with which we will communicate between
    # Shiny and the parallel coordinates without a re-render
    pcp <- parcoordsProxy("pc")

    pcHide(pcp, input$columns)
  }, ignoreInit = TRUE, ignoreNULL = FALSE)

  observeEvent(input$hidenames, {
    # create a proxy with which we will communicate between
    # Shiny and the parallel coordinates without a re-render
    pcp2 <- parcoordsProxy("pc2")
    if(input$hidenames) {

```

```

        pcHide(pcp2, "names")
      } else {
        pcUnhide(pcp2, "names")
      }
    })
  }

shinyApp(ui = ui, server = server)

### snapshot example ----
library(shiny)
library(parcoords)

ui <- tags$div(
  actionButton(inputId = "snapBtn", label = "snapshot"),
  parcoordsOutput("pc", height=400)
)

server <- function(input, output, session) {
  # create a proxy with which we will communicate between
  # Shiny and the parallel coordinates without a re-render
  pcp <- parcoordsProxy("pc")

  output$pc <- renderParcoords({
    parcoords(mtcars)
  })

  observeEvent(input$snapBtn, {
    # create a proxy with which we will communicate between
    # Shiny and the parallel coordinates without a re-render
    pcp <- parcoordsProxy("pc")
    pcSnapshot(pcp)
  })
}

shinyApp(ui=ui, server=server)
}

```

parcoordsProxy

Send commands to a Proxy instance in a Shiny app

Description

Creates a parcoords-like object that can be used to customize and control a parcoords that has already been rendered. For use in Shiny apps and Shiny docs only.

Usage

```
parcoordsProxy(parcoordsId, session = shiny::getDefaultReactiveDomain(),
  deferUntilFlush = TRUE)
```

Arguments

parcoordsId	single-element character vector indicating the output ID of the parcoords to modify (if invoked from a Shiny module, the namespace will be added automatically)
session	the Shiny session object to which the map belongs; usually the default value will suffice
deferUntilFlush	indicates whether actions performed against this instance should be carried out right away, or whether they should be held until after the next time all of the outputs are updated; defaults to TRUE

Details

Normally, you create a parcoords chart using the `parcoords` function. This creates an in-memory representation of a parcoords that you can customize. Such a parcoords can be printed at the R console, included in an R Markdown document, or rendered as a Shiny output.

In the case of Shiny, you may want to further customize a parcoords, even after it is rendered to an output. At this point, the in-memory representation of the parcoords is long gone, and the user's web browser has already realized the parcoords instance.

This is where `parcoordsProxy` comes in. It returns an object that can stand in for the usual parcoords object. The usual parcoords functions can be called, and instead of customizing an in-memory representation, these commands will execute on the live parcoords instance.

pcCenter	<i>Center parcoords horizontally based on column/variable through parcoordsProxy</i>
----------	--

Description

Center parcoords horizontally based on column/variable through `parcoordsProxy`

Usage

```
pcCenter(pc = NULL, dim = NULL)
```

Arguments

pc	<code>parcoordsProxy</code>
dim	string column/variable to center.

Value

`parcoords_proxy`

pcFilter	<i>Filter parcoords through parcoordsProxy</i>
----------	--

Description

Filter parcoords through parcoordsProxy

Usage

```
pcFilter(pc = NULL, filters = NULL)
```

Arguments

pc	parcoordsProxy
filters	list of filters to apply to the parcoords proxy. Please see search.js for example queries as filters.

Value

parcoords_proxy

pcHide	<i>Hide parcoords columns through parcoordsProxy</i>
--------	--

Description

Hide parcoords columns through parcoordsProxy

Usage

```
pcHide(pc = NULL, dim = NULL)
```

Arguments

pc	parcoordsProxy
dim	string column(s) to hide.

Value

parcoords_proxy

pcSnapshot	<i>Download image of parcoords through parcoordsProxy</i>
------------	---

Description

Download image of parcoords through parcoordsProxy

Usage

```
pcSnapshot(pc = NULL)
```

Arguments

pc	parcoordsProxy
----	----------------

Value

parcoords_proxy

pcUnhide	<i>Unhide parcoords columns through parcoordsProxy</i>
----------	--

Description

Unhide parcoords columns through parcoordsProxy

Usage

```
pcUnhide(pc = NULL, dim = NULL)
```

Arguments

pc	parcoordsProxy
dim	string column(s) to hide.

Value

parcoords_proxy

Index

JS, [3](#)

parcoords, [2](#), [9](#)

parcoords-shiny, [5](#)

parcoordsOutput (parcoords-shiny), [5](#)

parcoordsProxy, [8](#)

pcCenter, [9](#)

pcFilter, [10](#)

pcHide, [10](#)

pcSnapshot, [11](#)

pcUnhide, [11](#)

renderParcoords (parcoords-shiny), [5](#)