Package: sunburstR (via r-universe)

September 27, 2024

Type Package

Title Sunburst 'Htmlwidget'

Version 2.1.8

Date 2023-02-05

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

URL https://github.com/timelyportfolio/sunburstR

BugReports https://github.com/timelyportfolio/sunburstR/issues

Description Make interactive 'd3.js' sequence sunburst diagrams in R
with the convenience and infrastructure of an 'htmlwidget'.

License MIT + file LICENSE

LazyData TRUE

Imports d3r (>= 0.6.9), dplyr, htmlwidgets, htmltools

Suggests jsonlite, knitr, markdown, pipeR, testthat, tidyr (>= 0.7.0),
rmarkdown

Enhances treemap

RoxygenNote 7.2.3

VignetteBuilder knitr

Repository https://cjyetman.r-universe.dev

RemoteUrl https://github.com/timelyportfolio/sunburstR

RemoteRef HEAD

RemoteSha 0d70594a7b56a09b6ae66a2218c348e6f8bb3a80

Contents

add_shiny *Add Shiny Events*

Description

Add Shiny Events

Usage

```
add_shiny(sunburst = NULL)
```

Arguments

`sunburst`	`sunburst` htmlwidget to which you would like to add event handling

Value

`sunburst` htmlwidget

Examples

```
## Not run:

library(shiny)
library(sunburstR)

sequences <- read.csv(
  system.file("examples/visit-sequences.csv",package="sunburstR")
  ,header=F
  ,stringsAsFactors = FALSE
)


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

  output$sunburst <- renderSunburst({
    #invalidateLater(1000, session)

    sequences <- sequences[sample(nrow(sequences),1000),]

    add_shiny(sunburst(sequences))
  })


  selection <- reactive({
    input$sunburst_mouseover
  })

  output$selection <- renderText(selection())
}


ui<-fluidPage(
  sidebarLayout(
    sidebarPanel(

    ),

    # plot sunburst
    mainPanel(
      sunburstOutput("sunburst"),
      textOutput("selection")
    )
  )
)

shinyApp(ui = ui, server = server)

# an example with d2b sunburst and Shiny
library(shiny)
library(sunburstR)

# use a sample of the sequences csv data
sequences <- read.csv(
  system.file("examples/visit-sequences.csv",package="sunburstR")
  ,header = FALSE
  ,stringsAsFactors = FALSE
)[1:200,]

# create a d2b sunburst
s2b <- sund2b(sequences)

options(shiny.trace=TRUE)
ui <- sund2bOutput("s2b")
server <- function(input, output, session) {
  output$s2b <- renderSund2b({
    add_shiny(s2b)
  })
}
shinyApp(ui, server)

## End(Not run)
```

---

d2b-shiny *Shiny bindings for d2b*

Description

Output and render functions for using d2b within Shiny applications and interactive Rmd documents.

Usage

```
sund2bOutput(outputId, width = "100%", height = "400px")

renderSund2b(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 d2b
`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.

`sunburst` *'d3.js' Sequence Sunburst Diagrams*

Description

Sequences sunburst diagrams provide an interactive method of exploring sequence data, such as website navigation paths.

Usage

```
sunburst(
  data = NULL,
  legendOrder = NULL,
  colors = NULL,
  valueField = "size",
  percent = TRUE,
  count = FALSE,
  explanation = NULL,
  breadcrumb = list(),
  legend = list(),
  sortFunction = NULL,
  sumNodes = TRUE,
  withD3 = FALSE,
  width = NULL,
  height = NULL,
  elementId = NULL,
  sizingPolicy = NULL,
  csvdata = NULL,
  jsondata = NULL
)
```

Arguments

`data`	data in csv source,target form or in nested d3 JSON hierarchy with ‘name:..., children:[];‘. `csvdata` and `jsondata` arguments are now deprecated in favor of this single `data` argument. `list`, `character`, or `connection` data will be assumed to be `JSON`. `data.frame` data will be assumed to be `csvdata` and converted to `JSON` by `sunburstR:::csv_to_hier()`.
`legendOrder`	string vector if you would like to manually order the legend. If legendOrder is not provided, then the legend will be in the descending order of the top level hierarchy.
`colors`	`vector` of strings representing colors as hexadecimal for manual colors. If you want precise control of colors, supply a `list` with `range` and/or `domain`. For advanced customization, supply a JavaScript `function`.
`valueField`	`character` for the field to use to calculate size. The default value is `"size"`.
`percent`	`logical` to include percentage of total in the explanation.
`count`	`logical` to include count and total in the explanation.
`explanation`	JavaScript function to define a custom explanation for the center of the sunburst. Note, this will override `percent` and `count`.
`breadcrumb`	`list` to customize the breadcrumb trail. This argument should be in the form `list(w =, h =, s =, t = )` where `w` is the width, `h` is the height, `s` is the spacing, and `t` is the tail all in `px`. `w` is `0` by default for breadcrumbs widths based on text length.
`legend`	`list` to customize the legend or `logical` to disable the legend. The `list` argument should be in the form `list(w =, h =, r =, s = )` where `w` is the width, `h` is the height, `s` is the spacing, and `r` is the radius all in `px`.
`sortFunction`	`JS` function to sort the slices. The default sort is by size.
`sumNodes`	`logical` to sum non-leaf nodes. The default `sumNodes = TRUE` assumes that the user has not already calculated a sum.
`withD3`	`logical` to include d3 dependency from `d3r`. As of version 1.0, sunburst 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`.
`height, width`	height and width of sunburst htmlwidget containing div specified in any valid `CSS` size unit.
`elementId`	string id as a valid `CSS` element id.
`sizingPolicy`	see `sizingPolicy`.
`csvdata`	`deprecated` use data argument instead; data in csv source,target form
`jsondata`	`deprecated` use data argument instead; data in nested d3 JSON hierarchy with ‘name:..., children:[];‘

Examples

```
library(sunburstR)

# read in sample visit-sequences.csv data provided in source
# only use first 100 rows to speed package build and check
#   https://gist.github.com/kerryrodden/7090426#file-visit-sequences-csv
sequences <- read.csv(
  system.file("examples/visit-sequences.csv",package="sunburstR")
  ,header = FALSE
  ,stringsAsFactors = FALSE
)[1:100,]

sunburst(sequences)

## Not run:

# explore some of the arguments
sunburst(
  sequences
  ,count = TRUE
)

sunburst(
  sequences
  # apply sort order to the legends
  ,legendOrder = unique(unlist(strsplit(sequences[,1],"-")))
  # just provide the name in the explanation in the center
  ,explanation = "function(d){return d.data.name}"
)


# try with json data
sequence_json <- jsonlite::fromJSON(
  system.file("examples/visit-sequences.json",package="sunburstR"),
  simplifyDataFrame = FALSE
)
sunburst(sequence_json)



# try with csv data from this fork
#  https://gist.github.com/mkajava/7515402
# great use for new breadbrumb wrapping
sunburst(
  csvdata = read.csv(
    file = paste0(
      "https://gist.githubusercontent.com/mkajava/",
      "7515402/raw/9f80d28094dc9dfed7090f8fb3376ef1539f4fd2/",
      "comment-sequences.csv"
    )
    ,header = TRUE
    ,stringsAsFactors = FALSE
  )
)


# try with csv data from this fork
#  https://gist.github.com/rileycrane/92a2c36eb932b4f99e51/
sunburst( csvdata = read.csv(
  file = paste0(
    "https://gist.githubusercontent.com/rileycrane/",
    "92a2c36eb932b4f99e51/raw/",
    "a0212b4ca8043af47ec82369aa5f023530279aa3/visit-sequences.csv"
  )
  ,header=FALSE
  ,stringsAsFactors = FALSE
))

## End(Not run)
## Not run:
#  use sunburst to analyze ngram data from Peter Norvig
#    http://norvig.com/mayzner.html

library(sunburstR)
library(pipeR)

#  read the csv data downloaded from the Google Fusion Table linked in the article
ngrams2 <- read.csv(
  system.file(
    "examples/ngrams2.csv"
    ,package="sunburstR"
  )
  , stringsAsFactors = FALSE
)

ngrams2 %>>%
  #  let's look at ngrams at the start of a word, so columns 1 and 3
  (.[,c(1,3)]) %>>%
  #  split the ngrams into a sequence by splitting each letter and adding -
  (
    data.frame(
      sequence = strsplit(.[,1],"") %>>%
        lapply( function(ng){ paste0(ng,collapse = "-") } ) %>>%
        unlist
      ,freq = .[,2]
      ,stringsAsFactors = FALSE
    )
  ) %>>%
  sunburst


library(htmltools)

ngrams2 %>>%
  (
    lapply(
      seq.int(3,ncol(.))
      ,function(letpos){
        (.[,c(1,letpos)]) %>>%
          #  split the ngrams into a sequence by splitting each letter and adding -
          (
            data.frame(
              sequence = strsplit(.[,1],"") %>>%
                lapply( function(ng){ paste0(ng,collapse = "-") } ) %>>%
                unlist
              ,freq = .[,2]
              ,stringsAsFactors = FALSE
            )
          ) %>>%
          ( tags$div(style="float:left;",sunburst( ., height = 300, width = 300 )) )
      }
    )
  ) %>>%
  tagList %>>%
  browsable

## End(Not run)
## Not run:
  library(treemap)
  library(sunburstR)
  library(d3r)

  # use example from ?treemap::treemap
  data(GNI2014)
  tm <- treemap(GNI2014,
          index=c("continent", "iso3"),
          vSize="population",
          vColor="continent",
          type="index")

  tm_nest <- d3_nest(
    tm$tm[,c("continent", "iso3", "vSize", "color")],
    value_cols = c("vSize", "color")
  )

  sunburst(
    data = tm_nest,
    valueField = "vSize",
    count = TRUE,
    # to avoid double counting with pre-summed trees
    # use sumNodes = FALSE
    sumNodes = FALSE,
    colors = htmlwidgets::JS("function(d){return d3.select(this).datum().data.color;}"),
    withD3 = TRUE
  )

## End(Not run)
# calendar sunburst example

library(sunburstR)

df <- data.frame(
  date = seq.Date(
    as.Date('2014-01-01'),
    as.Date('2016-12-31'),
    by = "days"
  ),
  stringsAsFactors = FALSE
)

df$year = format(df$date, "%Y")
df$quarter = paste0("Q", ceiling(as.numeric(format(df$date,"%m"))/3))
df$month = format(df$date, "%b")
df$path = paste(df$year, df$quarter, df$month, sep="-")
df$count = rep(1, nrow(df))

sunburst(
  data.frame(xtabs(count~path,df)),
  # added a degree of difficulty by providing
  #  not easily sortable names
  sortFunction = htmlwidgets::JS(
"
function(a,b){
  abb = {
    2014:-7,
    2015:-6,
    2016:-5,
    Q1:-4,
    Q2:-3,
    Q3:-2,
    Q4:-1,
    Jan:1,
    Feb:2,
    Mar:3,
    Apr:4,
    May:5,
    Jun:6,
    Jul:7,
    Aug:8,
    Sep:9,
    Oct:10,
    Nov:11,
    Dec:12
  }
  return abb[a.data.name] - abb[b.data.name];
}
"
  )
)
# sorting example: place data in order of occurence

library(sunburstR)

df <- data.frame(
  group = c("foo", "bar", "xyz"),
  value = c(1, 3, 2)
```

```
)

sunburst(df,
         # create a trivial sort function
         sortFunction = htmlwidgets::JS('function(x) {return x;}'))

new_order <- c(3,2,1)
sunburst(df[new_order,],
         sortFunction = htmlwidgets::JS('function(x) {return x;}'))
```

sunburst-shiny *Shiny bindings for sunburst*

Description

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

Usage

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

renderSunburst(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.

sund2b *Sunburst Using 'd2b'*

Description

Create interactive sunburst chart with the 'd2b' charting library.

Usage

```
sund2b(
  data = NULL,
  colors = NULL,
  valueField = "size",
  tooltip = NULL,
  breadcrumbs = NULL,
  rootLabel = NULL,
  showLabels = FALSE,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

`data`	data in csv source,target form or in nested d3 JSON hierarchy with ‘name:..., children:[];‘. `list`, `character`, or `connection` data will be assumed to be `JSON`. `data.frame` data will be assumed to be `csvdata` and converted to `JSON` by `sunburstR:::csv_to_hier()`.
`colors`	`vector` of strings representing colors as hexadecimal for manual colors. If you want precise control of colors, supply a `list` with `range` and/or `domain`. For advanced customization, supply a JavaScript `function`.
`valueField`	`character` for the field to use to calculate size. The default value is `"size"`.
`tooltip`	`list` of options for customizing the tooltip. See the helper function `sund2bTooltip` for more information.
`breadcrumbs`	`list` of options for customizing the breadcrumb. See the helper function `sund2bBreadcrumb` for more information.
`rootLabel`	`character` to label root node something other than ’root’.
`showLabels`	`logical` to show labels on the slices. The default is `FALSE`.
`height, width`	height and width of sunburst htmlwidget containing div specified in any valid `CSS` size unit.
`elementId`	string id as a valid `CSS` element id.

Examples

```
if(interactive()){

  # The sund2b() API mirrors sunburst() with fewer arguments.

  library(sunburstR)

  # use a sample of the sequences csv data
  sequences <- read.csv(
    system.file("examples/visit-sequences.csv",package="sunburstR")
    ,header = FALSE
    ,stringsAsFactors = FALSE
  )[1:200,]

  # create a d2b sunburst
  sund2b(sequences)

  # show labels
  sund2b(sequences, showLabels = TRUE)

  # change the colors
  #   using d3.js categorical color scheme
  sund2b(
    sequences,
    colors = htmlwidgets::JS("d3.scaleOrdinal(d3.schemeCategory20b)")
  )
}

## Not run:
#  using RColorBrewer palette
sund2b(
  sequences,
  colors = list(range = RColorBrewer::brewer.pal(9, "Set3"))
)
#  using a color column from the R dataset
#  treemap has an amazing treecolors ability
library(treemap)
library(d3r)
rhd <- random.hierarchical.data()
tm <- treemap(
  rhd,
  index = paste0("index", 1:3),
  vSize = "x",
  draw = FALSE
)$tm
sund2b(
  d3_nest(tm, value_cols = colnames(tm)[-(1:3)]),
  colors = htmlwidgets::JS(
    # yes this is a little different, so please pay attention
    #  "function(d) {return d.color}" will not work
    "function(name, d){return d.color || '#ccc';}"
  ),
  valueField = "vSize"
)


# use sund2b in Shiny
library(shiny)
ui <- sund2bOutput("sun")
server <- function(input, output, session) {
  output$sun <- renderSund2b({
    sund2b(sequences)
  })
}
shinyApp(ui, server)
```

```
## End(Not run)
```

---

sund2bBreadcrumb *Advanced Customization of 'd2b' Breadcrumb*

Description

Advanced Customization of 'd2b' Breadcrumb

Usage

```
sund2bBreadcrumb(enabled = NULL, html = NULL, orient = NULL)
```

Arguments

`enabled`	`boolean` to enable or disable the breadcrumbs.
`html`	`character` or `htmlwidgets::JS` to customize the content of the breadcrumb. To provide a function, the arguments for the 'JavaScript' function will be 'function(nodedata, size, percent)' and the function should return a string.
`orient`	`character` which should be one of "top", "left", "right", "bottom" to control the orientation of the breadcrumb relative to the chart.

Value

`list`

Examples

```
if(interactive()){

library(sunburstR)

# use a sample of the sequences csv data
sequences <- read.csv(
  system.file("examples/visit-sequences.csv",package="sunburstR")
  ,header = FALSE
  ,stringsAsFactors = FALSE
)[1:200,]

# disable the breadcrumb
sund2b(
  sequences,
  breadcrumbs = sund2bBreadcrumb(
    enabled = FALSE
  )
)
# change the breadcrumb content
sund2b(
  sequences,
  breadcrumbs = sund2bBreadcrumb(
    html = htmlwidgets::JS("
function(nodedata, size, percent) {
  return '<span style=\"font-weight: bold;\">' + nodedata.name + '</span>' + ' ' + size
}
    ")
  )
)


}
```

`sund2bTooltip` *Advanced Customization of 'd2b' Tooltip*

Description

Advanced Customization of 'd2b' Tooltip

Usage

```
sund2bTooltip(at = NULL, followMouse = NULL, html = NULL, my = NULL)
```

Arguments

`at`	`character` which should be one of "top left", "top center", "top right", "center left", "center center", "center right", "bottom center", "bottom right" to specify where the tooltip will be positioned relative to the hovered item.
`followMouse`	`logical` controlling whether the tooltip will follow the mouse instead of being placed in a static position relative to the hovered element
`html`	`character` or `htmlwidgets::JS` to customize the content of the tooltip. To provide a function, the arguments for the 'JavaScript' function will be 'function(nodedata, size, percent)' and the function should return a string.
`my`	`character` which should be one of "top", "left", "right", "bottom" to control the orientation of the tooltip.

Value

`list`

Examples

```
if(interactive()){

library(sunburstR)

# use a sample of the sequences csv data
sequences <- read.csv(
  system.file("examples/visit-sequences.csv",package="sunburstR")
  ,header = FALSE
  ,stringsAsFactors = FALSE
)[1:200,]

# change the tooltip
sund2b(
  sequences,
  tooltip = sund2bTooltip(
    html = htmlwidgets::JS("
function(nodedata, size, percent) {
  return '<span style=\"font-weight: bold;\">' + nodedata.name + '</span>' + ' ' + size
}
    ")
  )
)


}
```

Index