Package ‘metricsgraphics’

December 21, 2015

Type Package

Title Create Interactive Charts with the JavaScript ‘MetricsGraphics’ Library

Version 0.9.0

Date 2015-12-21

Maintainer Bob Rudis <bob@rudis.net>

Description Provides an 'htmlwidgets' interface to the 'MetricsGraphics.js' ('D3'-based) charting library which is geared towards displaying time-series data. Chart types include line charts, scatterplots, histograms and rudimentary bar charts. Support for laying out multiple charts into a grid layout is also provided. All charts are interactive and many have an option for line, label and region annotations.

URL http://github.com/hrbrmstr/metricsgraphics

BugReports https://github.com/hrbrmstr/metricsgraphics/issues

License MIT + file LICENSE

Suggests testthat, RColorBrewer, ggplot2, ggplot2movies, jsonlite (>= 0.9.16), knitr (>= 1.8), shiny (>= 0.12.0), binom, dplyr, grDevices

Depends R (>= 3.0.0)

Imports magrittr, htmlwidgets, htmltools

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Author Bob Rudis [aut, cre], Ali Almossawi [ctb, cph] (MetricsGraphics library), Hamilton Ulmer [ctb, cph] (MetricsGraphics library), Mozilla [cph] (MetricsGraphics library), jQuery Foundation and contributors [ctb, cph] (jQuery library)

Repository CRAN

Date/Publication 2015-12-21 15:29:04
R topics documented:

metricsgraphics .................................................. 2
metricsgraphics-exports ......................................... 2
metricsgraphicsOutput ........................................... 3
mjs_add_baseline ................................................... 3
mjs_add_confidence_band ........................................ 4
mjs_add_css_rule .................................................. 5
mjs_add_legend ..................................................... 6
mjs_add_line ........................................................ 7
mjs_add_marker ..................................................... 8
mjs_add_mouseover ................................................ 8
mjs_annotate_region ............................................. 10
mjs_axis_x .......................................................... 11
mjs_axis_y .......................................................... 11
mjs_bar .............................................................. 12
mjs_grid .............................................................. 13
mjs_hist .............................................................. 14
mjs_histogram ..................................................... 15
mjs_labs ............................................................. 16
mjs_line .............................................................. 16
mjs_plot ............................................................. 17
mjs_point ............................................................ 19
renderMetricsgraphics ............................................ 20

Index .................................................................. 21

metricsgraphics An htmlwidget interface to the
Rhrefhttp://metricsgraphicsjs.org/MetricsGraphics.js D3 chart
library

Description

An htmlwidget interface to the MetricsGraphics.js D3 chart library

Author(s)

Bob Rudis (@hrbrmstr)

metricsgraphics-exports

metricsgraphics exported operators

Description

The following functions are imported and then re-exported from the dygraphs package to enable use of the magrittr pipe operator with no additional library calls
metricsgraphicsOutput  Widget output function for use in Shiny

Description

Widget output function for use in Shiny

Usage

metricsgraphicsOutput(outputId, width = "100\%", height = "400px")

Arguments

outputId  output id
width  width
height  height

mjs_add_baseline  Sets a baseline line/label

Description

metricsgraphics baselines are horizontal lines that may specify, say, a goal or target to be reached. This function lets you add baselines to a plot object. you can add as many as you need to.

Usage

mjs_add_baseline(mjs, y_value, label)

Arguments

mjs  metricsgraphics plot object
y_value  which y value to draw the baseline at
label  text label for the marker

Value

metricsgraphics object
mjs_add_confidence_band

Examples

```r
data.frame(
    year=seq(1790, 1970, 10),
    uspop=as.numeric(uspop)
) %>%
mjs_plot(x=year, y=uspop) %>%
mjs_line() %>%
mjs_add_marker(1850, "Something Wonderful") %>%
mjs_add_baseline(150, "Something Awful")
```

Description

If you have lower & upper points associated with your line in a data frame, you can specify their accessors (defaults to "l" & "u") here which will result in a shaded confidence band being plotted with the line.

Usage

```r
mjs_add_confidence_band(mjs, lower_accessor = "l", upper_accessor = "u")
```

Arguments

- `mjs` metricsgraphics plot object
- `lower_accessor` bare or quoted name of column to use for the lower bound of the confidence band
- `upper_accessor` bare or quoted name of column to use for the upper bound of the confidence band

Examples

```r
require(binom)
require(dplyr)

set.seed(1492)
binom.confint(x=sample(2:30, 100, replace=TRUE), n = 100, tol = 1e-8, 
    methods="bayes") %>%
mutate(x=1:100) -> bdat

bdat %>%
mjs_plot(x=x, y=mean, width=600, height=240) %>%
mjs_axis_x(show_secondary_x_label=FALSE, 
    extended_ticks=TRUE) %>%
mjs_line() %>%
mjs_add_confidence_band(lower_accessor="lower", 
    upper_accessor="upper")
```
mjs_add_css_rule

Add a CSS rule to the rendered htmlwidget

Description
This function will add a CSS rule to a widget-created DOM stylesheet. rule should be a valid CSS rule as you would enter in a <style>...</style> block. No checking is done to ensure validity.

Usage
mjs_add_css_rule(mjs, rule, warn = TRUE)

Arguments

mjs  metricsgraphics plot object
rule  character vector of CSS rule(s) to add to the widget DOM
warn  show warnings for global CSS rules? (default: TRUE)

Details
Use {{ID}} (followed by a space) to target the CSS rule just to the widget vs the whole DOM. Vectorized over rule

Value
metricsgraphics plot object

Note
This is for expert use only. You need to know quite a bit about the visualization and target DOM to effectively use this function. CSS rules without the {{ID}} are applied to the entire DOM.

Examples

set.seed(1492)
stocks <- data.frame(
  time = as.Date('2009-01-01') + (365 * 0:9),
  X = rnorm(10, 0, 1),
  Y = rnorm(10, 0, 2),
  Z = rnorm(10, 0, 4))
stocks %>%
mjs_plot(x=time, y=X) %>%
mjs_line() %>%
mjs_axis_x(xax_format="date") %>%
mjs_add_css_rule("{{ID}} .blk { fill:black }") %>%
mjs_annotate_region("2013-01-01", "2016-01-01", "Crazy times", "blk")
mjs_add_legend

Adds a legend to a metricsgraphics chart

Description

Adds a legend to a metricsgraphics chart

Usage

mjs_add_legend(mjs, legend, inline = FALSE)

Arguments

- mjs: metricsgraphics plot object
- legend: character vector of labels for the legend
- inline: TRUE if you want line labels to the right of the chart vs in a legend block (experimental)

Value

metricsgraphics object

Examples

```r
set.seed(1492)
stocks <- data.frame(
  time = as.Date('2009-01-01') + 0:9,
  X = rnorm(10, 0, 1),
  Y = rnorm(10, 0, 2),
  Z = rnorm(10, 0, 4))

stocks %>%
mjs_plot(x=time, y=X) %>%
mjs_line() %>%
mjs_add_line(Y) %>%
mjs_add_line(Z) %>%
mjs_axis_x(xax_format="date") %>%
mjs_add_legend(legend=c("X", "Y", "Z"))
```
mjs_add_line

Add a new line to a metricsgraphics.js linechart "geom"

Description

This function adds a line to an existing mjs_line "geom". Specify the bare or quoted name of the column to use in y_accessor and it will be added to the plot.

Usage

mjs_add_line(mjs, y_accessor, color = NULL)

Arguments

mjs metricsgraphics plot object
y_accessor bare or quoted name of column to add to the existing line plot
color line color. Use NULL (the default) to use default Metrics Graphics colors or if you plan on using CSS to control the colors.

Value

metricsgraphics object

Note

You must have called mjs_line first before adding additional columns. If you plan on using cusom colors, all lines must have a color value or the result is non-deterministic.

Examples

set.seed(1492)
stocks <- data.frame(
  time = as.Date('2009-01-01') + 0:9,
  X = rnorm(10, 0, 1),
  Y = rnorm(10, 0, 2),
  Z = rnorm(10, 0, 4))

stocks %>%
  mjs_plot(x=time, y=X) %>%
  mjs_line() %>%
  mjs_add_line(Y) %>%
  mjs_add_line(Z) %>%
mjs_axis_x(xax_format="date")
mjs_add_marker

*Sets a marker line/label*

**Description**

metricsgraphics marker lines are vertical lines that identify, say, events or dates worth annotating. This function lets you add a marker to a plot object. You can add as many as you need to.

**Usage**

```r
mjs_add_marker(mjs, x_value, label)
```

**Arguments**

- `mjs`: metricsgraphics plot object
- `x_value`: which x value to draw the marker at
- `label`: text label for the marker

**Value**

metricsgraphics object

**Examples**

```r
data.frame(
  year=seq(1790, 1970, 10),
  uspop=as.numeric(uspop)
) 

mjs_plot(x=year, y=uspop) 

mjs_line() 

mjs_add_marker(1850, "Something Wonderful") 

mjs_add_baseline(150, "Something Awful")
```

mjs_add_mouseover

*Adds a custom rollover to a metricsgraphics chart*

**Description**

MetricsGraphics charts allow for custom rollovers. `mjs_add_mouseover` lets you add a custom rollover to a metricsgraphics object. You must be familiar with javascript and D3 idioms since you are supplying a javascript function as a parameter.

Since targeting is done by element id, you will need to add a special string - {{ID}} - to the target element selector so metricsgraphics can add the unique object identifier to the selector. See Examples for basic usage.
**mjs_add_mouseover**

**Usage**

```r
mjs_add_mouseover(mjs, func)
```

**Arguments**

- `mjs` metricsgraphics plot object
- `func` text for javascript function to be used for the custom rollover. See Details for usage.

**Value**

metricsgraphics object

**Note**

you need to use `dNpointNTHING` vs `dNTHING` when trying to add mouseovers to a metricsgraphics scatterplot.

**Examples**

```r
data <- data.frame(date = as.Date('2009-01-01') + 0:9,
                   value = rnorm(10, 0, 2))
data <- data.frame(value = rnorm(n=30, mean=5, sd=1),
                   value2=rnorm(n=30, mean=4, sd=1),
                   test = c(rep(c('test', 'test2'), 15)))
data <- data.frame(x = value, y = value2)
data <- data.frame(x = date, y = value)
```

```r
mjs_plot(x=date, y=value) mjs_line() mjs_axis_x(xax_format = "date") mjs_add_mouseover("function(d, i) {
  d
  i
}")
mjs_add_mouseover("function(d, i) {
  d
  i
}")
```

# slightly different for scatterplots

```r
mjs_plot(x=date, y=value) mjs_point() mjs_add_mouseover("function(d, i) {
  d
  i
}")
mjs_add_mouseover("function(d, i) {
  d
  i
}")
```
Description

This function uses the mg-regions plugin to enable region highlighting with an optional label.

Usage

mjs_annotate_region(mjs, x_start = NULL, x_end = NULL, label = NULL, css_class = NULL)

Arguments

mjs metricsgraphics object
x_start start point on x axis for region annotation
x_end end point on x axis for region annotation
label text label for annotation (leave NULL for no label
css_class CSS class to apply (see References link for more information)

Details

This function is also experimental and relies on the plugin maintainer to continue support for it. You should be well-versed in CSS to use this function properly.

Value

metricsgraphics object

References

https://github.com/senseyeio/mg-regions

Examples

data.frame(year=seq(1790, 1790, 10),
            uspop=as.numeric(uspop)) #>>
mjs_plot(x=year, y=uspop, title="Population Chart") #>>
mjs_line() #>>
mjs_annotate_region(1850, 1900, "Bad stuff") #>>
mjs_annotate_region(1810, 1830, "Stuff")

set.seed(1492)
stocks <- data.frame(
   time = as.Date('2009-01-01') + (365 * 0:9),
   X = rnorm(10, 0, 1),
   Y = rnorm(10, 0, 2),
   Z = rnorm(10, 0, 4))

mjs_annotate_region
mjs_axis_x

## stocks

```r
mjs_plot(y = X)
mjs_line()
```

```r
mjs_axis_x(xax_format = "date")
mjs_annotate_region("2013-01-01", "2016-01-01", "Crazy times")
```

```r
mjs_plot(x = time, y = X)
mjs_line()
```

```r
mjs_axis_x(xax_format = "date")
mjs_add_css_rule(".blk { fill: black }")
mjs_annotate_region("2013-01-01", "2016-01-01", "Crazy times", "blk")
```

---

### mjs_axis_x

**Configure x axis ticks & limits**

**Description**

Configure x axis ticks & limits

**Usage**

```r
mjs_axis_x(mjs, show = TRUE, xax_count = 6, min_x = NULL, max_x = NULL,
extended_ticks = FALSE, xax_format = "plain",
show_secondary_x_label = NULL, rug = FALSE)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mjs</td>
<td>metricsgraphics plot object</td>
</tr>
<tr>
<td>show</td>
<td>display the axis? (default: TRUE - yes)</td>
</tr>
<tr>
<td>xax_count</td>
<td>tick count</td>
</tr>
<tr>
<td>min_x</td>
<td>min limit for x axis</td>
</tr>
<tr>
<td>max_x</td>
<td>max limit for x axis</td>
</tr>
<tr>
<td>extended_ticks</td>
<td>extend ticks on x axis?</td>
</tr>
<tr>
<td>xax_format</td>
<td>how to format tick labels. Currently one of &quot;plain&quot;, &quot;comma&quot; or &quot;date&quot;</td>
</tr>
<tr>
<td>show_secondary_x_label</td>
<td>determines whether to show the year, or another unit of time in the case of smaller series, on the x-axis below the x-axis labels.</td>
</tr>
<tr>
<td>rug</td>
<td>show a &quot;rug&quot; plot next to the x axis? (default: FALSE - no)</td>
</tr>
</tbody>
</table>

**Note**

xax_format is likely to undergo a drastic change in future releases but support for these three formats will also likely remain.
mjs_axis_y  

Configure y axis ticks & limits

Description
Configure y axis ticks & limits

Usage
mjs_axis_y(mjs, show = TRUE, yax_count = 5, min_y = NULL, max_y = NULL,
extended_ticks = FALSE, y_scale_type = "linear", yax_units = ",",
rug = FALSE)

Arguments
mjs  
metricsgraphics plot object

show  
display the axis? (default: TRUE - yes)

yax_count  
tick count

min_y  
min limit for y axis

max_y  
max limit for y axis

extended_ticks  
extend ticks on y axis?

y_scale_type  
scale for y axis; either "linear" (default) or "log"

yax_units  
a prefix symbol to be shown alongside the y axis' labels. Useful for currencies, for instance.

rug  
show a "rug" plot next to the y axis? (default: FALSE - no)

Value
metricsgraphics object

mjs_bar  
metricsgraphics.js bar chart "geom"

Description
This function adds a bar "geom" to a metricsgraphics.js html widget.

Usage
mjs_bar(mjs, bar_height = 20, binned = TRUE)
### mjs_grid

**Description**

Lays out metricsgraphics widgets into a "grid", similar to `grid.arrange` from `gridExtra`

**Usage**

```
mjs_grid(..., ncol = 1, nrow = 1, widths = 1)
```

**Arguments**

- `...` either individual `metricsgraphics` objects or a mixture of individual `metricsgraphics` objects and lists of `metricsgraphics` objects.
- `ncol` how many columns in the grid
- `nrow` how many rows in the grid
- `widths` widths of the cells per row

**Value**

`htmltools` tag with wrapped `metricsgraphics` objects suitable for knitting with `echo=FALSE` & `results="asis"` or rendering in Viewer with `html_print`

---

### Arguments

- `mjs` `metricsgraphics` plot object
- `bar_height` width of bars
- `binned` is data already binned? (default: `TRUE - yes`)

**Value**

`metricsgraphics` object

**Note**

`metricsgraphics.js` currently has "meh" support for bar charts

**Examples**

```
data.frame(year=seq(1790, 1970, 10),
            uspop=as.numeric(uspop)) %>%
mjs_plot(x=year, y=uspop, width=300, height=400) %>
mjs_bar()
```
Note

mjs_grid does not work in a Shiny context

---

`mjs_hist`  *Shortcut for plotting MetricsGraphics histograms*

Description

This function performs the call to `mjs_plot` and assumes `data` is a numeric vector. It's intended to save keystrokes when plotting quick histograms. This function automatically a y axis label “Frequency” which you can override with a call to `mjs_labs`.

Usage

```r
mjs_hist(data, bins = NULL, bar_margin = 1)
```

Arguments

- `data` numeric vector
- `bins` number of bins for the histogram (NULL == let MetricsGraphics.js library compute)
- `bar_margin` space between bars (defaults to 1)

Value

metricsgraphics object

Examples

```r
bimod <- c(rnorm(1000, 0, 1), rnorm(1000, 3, 1))
mjs_plot(bimod) %>% mjs_histogram()
```

```r
bimod %>% mjs_hist()
```

```r
mjs_plot(bimod) %>% mjs_histogram(bins=30)
```

```r
bimod %>% mjs_hist(30)
```
mjs_histogram

Plot Histograms with MetricsGraphics

Description

Given a numeric vector or a data frame and numeric column name (bare or quoted), plot a histogram with the specified parameter. This function automatically a y axis label "Frequency" which you can override with a call to mjs_labs.

Usage

mjs_histogram(mjs, bar_margin = 1, bins = NULL)

Arguments

mjs       metricsgraphics plot object
bar_margin space between bars (defaults to 1)
bins      number of bins for the histogram (NULL == let MetricsGraphics.js library compute)

Value

metricsgraphics plot object

Examples

movies <- ggplot2movies::movies[sample(nrow(ggplot2movies::movies), 1000), ]
mjs_plot(movies$rating) %>% mjs_histogram()
mjs_plot(movies, rating) %>%
  mjs_histogram() %>%
  mjs_labs(x_label="Histogram of movie ratings")
mjs_plot(movies$rating) %>%
  mjs_histogram(bins=30)
mjs_plot(runif(10000)) %>%
  mjs_histogram() %>%
  mjs_labs(x_label="runif(10000)")
mjs_labs Configure axis labels & plot description

Description
Configure axis labels & plot description

Usage
mjs_labs(mjs, x_label = NULL, y_label = NULL)

Arguments
mjs metricsgraphics object
x_label label for x axis
y_label label for y axis

Value
metricsgraphics object

Examples
mtcars %>%
mjs_plot(x=wt, y=mpg, width=400, height=300) %>%
mjs_point(color_accessor=carb, size_accessor=carb) %>%
mjs_labs(x="Weight of Car", y="Miles per Gallon")

mjs_line metricsgraphics.js linechart "geom"

Description
This function adds a line "geom" to a metricsgraphics.js html widget.

Usage
mjs_line(mjs, area = FALSE, animate_on_load = FALSE, color = NULL, interpolate = "cardinal")
Arguments

- **mjs**: metricsgraphics plot object
- **area**: fill in area under line? (default: FALSE - no)
- **animate_on_load**: animate the drawing of the plot on page load? (default: FALSE - no)
- **color**: line color (hex string or valid HTML color string). Use NULL (the default) to use the default Metrics Graphics colors or if you plan on controlling the colors with CSS.

Value

metricsgraphics object

Note

If you plan on using custom colors, all lines must have a color value or the result is non-deterministic.

Examples

```r
data.frame(year=seq(1790, 1970, 10),
           uspop=as.numeric(uspop)) %>%
mjs_plot(x=year, y=uspop) %>%
mjs_line()
```

---

**mjs_plot**

Create a new metricsgraphics.js plot

Description

*mjs_plot()* initializes the metricsgraphics.js html widget and takes a data frame & (bare or quoted) x & y column names as minimum input. This must be piped to a "geom" (metricsgraphics.js only supports single "geom" layers) and can also be piped to other mjs_ functions that manipulate aesthetics.

Usage

```r
mjs_plot(data, x, y, show_rollover_text = TRUE, linked = FALSE,
          decimals = 2, format = "count", missing_is_hidden = FALSE, left = 80,
          right = 10, top = 40, bottom = 60, buffer = 8, width = NULL,
          height = NULL, title = NULL, description = NULL)
```
Arguments

data 
  data frame  
x  
  bare or quoted name of column to use for x values  
y  
  bare or quoted name of column to use for y values  
show_rollover_text  
  determines whether or not to show any text when a data point is rolled over.  
linked  
  inks together all other graphs whose linked option is set to true. When one graphic in that set is rolled over, the corresponding values in the other graphics are also rolled over (default: FALSE - not linked)  
decimals  
  the number of decimals to show in a rollover (default: 2)  
format  
  sets the format of the data object, which is to say, counts or percentages  
missing_is_hidden  
  if true and if the data object is a time series, missing data points will be treated as zeros  
left  
  the size of the left margin in pixels.  
right  
  the size of the right margin in pixels.  
top  
  the size of the top margin in pixels.  
bottom  
  the size of the bottom margin in pixels.  
buffer  
  the buffer size in pixels between the actual chart area and the margins.  
width  
  Width in pixels (optional, defaults to automatic sizing)  
height  
  Height in pixels (optional, defaults to automatic sizing)  
title  
  plot title  
description  
  plot description

Details

See MetricsGraphics.js for more information.

Value

metricsgraphics object

Note

Plot title and description work best when the widget is in a Bootstrap template. They also increase the overall plot area (height, mostly) since they add <div>s. The description will be visible in the upper left area (on hover) if not displayed in a Bootstrap template.
mjs_point

Examples

data.frame(year=seq(1790, 1970, 10),
  uspop=as.numeric(uspop)) %>%
mjs_plot(x=year, y=uspop) %>%
mjs_line()

# accessor params can also be quoted

data.frame(year=seq(1790, 1970, 10),
  uspop=as.numeric(uspop)) %>%
mjs_plot(x="year", y="uspop") %>%
mjs_line()

---

mjs_point | metricsgraphics.js scatterplot "geom"

Description

This function adds a point/scatterplot "geom" to a metricsgraphics.js html widget.

Usage

mjs_point(mjs, point_size = 2.5, least_squares = FALSE,
  size_accessor = NULL, color_accessor = NULL, color_type = "number",
  color_range = c("blue", "red"), size_range = c(1, 5), x_rug = FALSE,
  y_rug = FALSE)

Arguments

- **mjs**: metricsgraphics plot object
- **point_size**: the radius of the dots in the scatterplot
- **least_squares**: add a least squares line? (default: FALSE - no)
- **size_accessor**: bare or quoted name of a column to use to scale the size of the points
- **color_accessor**: bare or quoted name of a column to use to scale the color of the points
- **color_type**: specifies whether the color scale is quantitative or qualitative. By setting this option to "category", you can color the points according to some other discrete value
- **color_range**: the range of colors, used to color different groups of points.
- **size_range**: specifies the range of point sizes, when point sizes are mapped to data
- **x_rug**: show a "rug" plot next to the x axis? (default: FALSE - no)
- **y_rug**: show a "rug" plot next to the y axis? (default: FALSE - no)

Value

metricsgraphics object
Examples

mtcars %>%
  mjs_plot(x=wt, y=mpg, width=400, height=300) %>%
  mjs_point(least_squares=TRUE)

---

renderMetricsgraphics  Widget render function for use in Shiny

Description

Widget render function for use in Shiny

Usage

renderMetricsgraphics(expr, env = parent.frame(), quoted = FALSE)

Arguments

expr  expr
env   env
quoted quoted
Index

`%>% (metricsgraphics-exports), 2`

`JS (metricsgraphics-exports), 2`

`metricsgraphics, 2`
`metricsgraphics-exports, 2`
`metricsgraphics-package`
  `(metricsgraphics), 2`
`metricsgraphicsOutput, 3`
`mjs_add_baseline, 3`
`mjs_add_confidence_band, 4`
`mjs_add_css_rule, 5`
`mjs_add_legend, 6`
`mjs_add_line, 7`
`mjs_add_marker, 8`
`mjs_add_mouseover, 8`
`mjs_annotate_region, 10`
`mjs_axis_x, 11`
`mjs_axis_y, 12`
`mjs_bar, 12`
`mjs_grid, 13`
`mjs_hist, 14`
`mjs_histogram, 15`
`mjs_labs, 16`
`mjs_line, 16`
`mjs_plot, 17`
`mjs_point, 19`

`renderMetricsgraphics, 20`

`saveWidget (metricsgraphics-exports), 2`