Package ‘billboarder’

March 27, 2021

Title Create Interactive Chart with the JavaScript 'Billboard' Library

Version 0.3.1

Description Provides an 'htmlwidgets' interface to 'billboard.js',
a re-usable easy interface JavaScript chart library, based on D3 v4+.
Chart types include line charts, scatterplots, bar/lollipop charts, histogram/density plots, pie/donut charts and gauge charts.
All charts are interactive, and a proxy method is implemented to smoothly update a chart without rendering it again in 'shiny' apps.

URL https://github.com/dreamRs/billboarder

BugReports https://github.com/dreamRs/billboarder/issues

Depends R (>= 3.1.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports htmlwidgets, htmltools, magrittr, jsonlite, ggplot2, scales, shiny, rlang

Suggests RColorBrewer, testthat, knitr, rmarkdown, covr

VignetteBuilder knitr

NeedsCompilation no

Author Victor Perrier [aut, cre],
Fanny Meyer [aut],
NAVER Corp [cph] (billboard.js library),
Mike Bostock [cph] (d3.format library)

Maintainer Victor Perrier <victor.perrier@dreamrs.fr>

Repository CRAN

Date/Publication 2021-03-27 21:50:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>billboarder-package</td>
<td>3</td>
</tr>
<tr>
<td>avengers</td>
<td>4</td>
</tr>
<tr>
<td>bauge</td>
<td>4</td>
</tr>
<tr>
<td>bauge-shiny</td>
<td>6</td>
</tr>
<tr>
<td>bb_add_style</td>
<td>7</td>
</tr>
<tr>
<td>bb_area</td>
<td>8</td>
</tr>
<tr>
<td>bb_axis</td>
<td>9</td>
</tr>
<tr>
<td>bb_bar</td>
<td>10</td>
</tr>
<tr>
<td>bb_barchart</td>
<td>10</td>
</tr>
<tr>
<td>bb_bar_color_manual</td>
<td>12</td>
</tr>
<tr>
<td>bb_bubble</td>
<td>13</td>
</tr>
<tr>
<td>bb_callbacks</td>
<td>14</td>
</tr>
<tr>
<td>bb_categories</td>
<td>15</td>
</tr>
<tr>
<td>bb_color</td>
<td>16</td>
</tr>
<tr>
<td>bb_colors_manual</td>
<td>17</td>
</tr>
<tr>
<td>bb_data</td>
<td>18</td>
</tr>
<tr>
<td>bb_densityplot</td>
<td>19</td>
</tr>
<tr>
<td>bb_donut</td>
<td>20</td>
</tr>
<tr>
<td>bb_donutchart</td>
<td>21</td>
</tr>
<tr>
<td>bb_export</td>
<td>22</td>
</tr>
<tr>
<td>bb_gauge</td>
<td>23</td>
</tr>
<tr>
<td>bb_gaugechart</td>
<td>24</td>
</tr>
<tr>
<td>bb_grid</td>
<td>25</td>
</tr>
<tr>
<td>bb_histogram</td>
<td>26</td>
</tr>
<tr>
<td>bb_interaction</td>
<td>28</td>
</tr>
<tr>
<td>bb_labs</td>
<td>29</td>
</tr>
<tr>
<td>bb_legend</td>
<td>30</td>
</tr>
<tr>
<td>bb_line</td>
<td>31</td>
</tr>
<tr>
<td>bb_linechart</td>
<td>31</td>
</tr>
<tr>
<td>bb_load</td>
<td>34</td>
</tr>
<tr>
<td>bb_lollipop</td>
<td>35</td>
</tr>
<tr>
<td>bb_pie</td>
<td>36</td>
</tr>
<tr>
<td>bb_piechart</td>
<td>37</td>
</tr>
<tr>
<td>bb_point</td>
<td>38</td>
</tr>
<tr>
<td>bb_proxy_axis_labels</td>
<td>39</td>
</tr>
<tr>
<td>bb_proxy_data_colors</td>
<td>39</td>
</tr>
<tr>
<td>bb_proxy_data_names</td>
<td>41</td>
</tr>
<tr>
<td>bb_proxy_flow</td>
<td>42</td>
</tr>
<tr>
<td>bb_proxy_focus</td>
<td>44</td>
</tr>
<tr>
<td>bb_proxy_groups</td>
<td>45</td>
</tr>
<tr>
<td>bb_proxy_hide</td>
<td>46</td>
</tr>
<tr>
<td>bb_proxy_legend</td>
<td>46</td>
</tr>
<tr>
<td>bb_proxy_show</td>
<td>48</td>
</tr>
<tr>
<td>bb_proxy_tooltip</td>
<td>49</td>
</tr>
<tr>
<td>bb_proxy_transform</td>
<td>50</td>
</tr>
<tr>
<td>bb_proxy_xs</td>
<td>50</td>
</tr>
</tbody>
</table>
## Description

This package allows you to use billboard.js ([https://naver.github.io/billboard.js/](https://naver.github.io/billboard.js/)), a reusable easy interface JavaScript chart library, based on D3 v4+.

## Author(s)

Victor Perrier (@dreamRs_fr)
avengers  
*Power ratings for The Avengers.*

**Description**
Data are available in "long" and "wide" format.

**Usage**

```r
avengers

avengers_wide
```

**Format**
A data frame with 24 rows and 4 variables:

- **group** Name of the hero
- **axis** Power skill
- **value** Value (1-7)
- **description** Character description

An object of class `data.frame` with 6 rows and 5 columns.

**Source**
Marvel Wikia (https://marvel.fandom.com/wiki/Marvel_Database) and Chris Zhou (http://bl.ocks.org/chrisrzhou/2421ac6541b68c1680f8)

---

bauge  
*Simple Gauge*

**Description**
A gauge that automatically updates itself in Shiny apps.

**Usage**

```r
bauge(
  value,
  min = 0,
  max = 100,
  colors = NULL,
  steps = NULL,
  label_tooltip = NULL,
)```
bauge

label_show = TRUE,
label_format = NULL,
label_extents = NULL,
expand = TRUE,
subtitle = NULL,
full_circle = FALSE,
gauge_width = NULL,
width = NULL,
height = NULL,
elementId = NULL
)

Arguments

value  Value for the gauge.
min    Minimal value for the gauge, default to 0.
max    Maximal value for the gauge, default to 100.
colors Vector of color(s), if more than one, steps must be specified.
steps  Upper bound for changing colors.
label_tooltip Label to appear on the tooltip, when mouse is hovering the gauge.
label_show Show or not minimal and maximal labels.
label_format JavaScript function to format inner label.
label_extents JavaScript function to set custom labels.
expand Enable or disable expanding gauge.
subtitle Additional text to add below the value.
full_circle Show full circle as donut. When set to TRUE, the max label will not be showed due to start and end points are same location.
gauge_width Set width of gauge chart.
width    Width of the element container.
height   Height of the element container.
elementId Use an explicit element ID for the widget.

Examples

bauge(45)

bauge(67, colors = "#F6C600")

bauge(90, full_circle = TRUE)

bauge(90, max = 210, gauge_width = 20, label_format = suffix(" km/h"))

# Shiny example
if (interactive()) {

library(shiny)

ui <- fluidPage(
  baugeOutput(outputId = "gauge", width = "300px"),
  actionButton(inputId = "update_value", label = "Update value"),
  actionButton(inputId = "update_max", label = "Update max"),
)

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

  value <- reactive({
    input$update_value
    round(sample.int(100, 1))
  })

  max_value <- reactive({
    input$update_max
    sample(100:200, 1)
  })

  output$gauge <- renderBauge({
    bauge(
      value = value(),
      max = max_value(),
      steps = c(30, 60, 90, 100),
      colors = c("#FF0000", "#F97600", "#F6C600", "#60B044")
    )
  })

  shinyApp(ui, server)
}

---

**bauge-shiny**  
**Shiny bindings for bauge**

**Description**

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

**Usage**

baugeOutput(outputId, width = "100\%", height = "200px")

renderBauge(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 bauge.
- **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.

---

**bb_add_style**

*Add custom style for regions and grid lines*

**Description**

Add custom style for regions and grid lines

**Usage**

```r
bb_add_style(
    bb,
    region = NULL,
    x_grid = NULL,
    y_grid = NULL,
    ..., .list = NULL
)
```

**Arguments**

- **bb**: A billboard htmlwidget object.
- **region**: A named list with style associated with region.
- **x_grid**: A named list with style associated with grid line on the X-axis.
- **y_grid**: A named list with style associated with grid line on the Y-axis.
- **..., .list**: Used internally.

**Value**

A billboard htmlwidget object.
Examples

# Change default color for regions
billboarder() %>%
  bb_linechart(data = sin(seq(-pi, pi, length.out = 30))) %>%
  bb_regions(
    list(start = 0, end = 10, class = "custom"), # add custom class
    list(start = 19, end = 29, class = "foo")
  ) %>%
  bb_add_style(region = list(custom = "fill: red;", foo = "fill: #009246;"))

# Customize grid line and text
billboarder() %>%
  bb_linechart(data = sin(seq(-pi, pi, length.out = 30))) %>%
  bb_y_grid(lines = list(list(
    value = 0, text = "Zero", position = "middle", class = "zero"
  ))) %>%
  bb_add_style(y_grid = list(
    zero = list(line = "stroke: red", text = "font-size: 240%; fill: black"
  )))

bb_area

Area property for a Billboard.js chart

Description

Area property for a Billboard.js chart

Usage

bb_area(bb, ...)

Arguments

bb A billboard htmlwidget object.

See https://naver.github.io/billboard.js/release/latest/doc/Options.html#area

Value

A billboard htmlwidget object.
bb_axis

Add axis parameters

Description
Add axis parameters

Usage
bb_axis(bb, ...)
bb_x_axis(bb, ...)
bb_y_axis(bb, ...)

Arguments
bb A billboard htmlwidget object.
...
Arguments defined in https://naver.github.io/billboard.js/demo/.

Value
A billboard htmlwidget object.

Examples

stars <- data.frame(
    package = c("billboarder", "ggriraph", "officer", "shinyWidgets", "visNetwork"),
    stars = c(9, 178, 43, 46, 175)
)

# Add a label to y axis
billboarder() %>%
    bb_barchart(data = stars) %>%
    bb_axis(y = list(label = list(text = "# of stars", position = "middle")))

# or shorter :
billboarder() %>%
    bb_barchart(data = stars) %>%
    bb_y_axis(label = list(text = "# of stars", position = "outer-top"))
**bb_bar**

*Bar property for a Billboard.js chart*

**Description**

Bar property for a Billboard.js chart

**Usage**

`bb_bar(bb, ...)`

**Arguments**

- `bb`: A billboard htmlwidget object.
- `...`: See https://naver.github.io/billboard.js/release/latest/doc/Options.html#bar

**Value**

A billboard htmlwidget object.

**Examples**

```r
billboader() %>%
  bb_barchart(data = data.frame(v1 = c("a", "b", "c"), value = c(5, 6, 3))) %>%
  bb_bar(width = list(ratio = 0.95))
```

---

**bb_barchart**

*Helper for creating a bar chart*

**Description**

Helper for creating a bar chart

**Usage**

```r
bb_barchart(
  bb,
  data,
  mapping = NULL,
  stacked = FALSE,
  rotated = FALSE,
  color = NULL,
  ...
)
```
Arguments

- **bb**: A billboard htmlwidget object.
- **data**: A data.frame, the first column will be used for x axis unless specified otherwise in mapping. If not a data.frame, an object coercible to data.frame.
- **mapping**: Mapping of variables on the chart, see bbaes.
- **stacked**: Logical, if several columns are provided, produce a stacked bar chart, else a dodge bar chart.
- **rotated**: Switch x and y axis position.
- **color**: Bar's color.
- **...**: Arguments for slot bar, see https://naver.github.io/billboard.js/release/latest/doc/Options.html#.bar.

Value

A billboard htmlwidget object.

Note

This function can be used with billboarderProxy in shiny application.

Examples

```r
stars <- data.frame(
  package = c("billboarder", "ggiraph", "officer",
                "shinyWidgets", "visNetwork", "rAmCharts",
                "D3partitionR"),
  stars = c(67, 252, 160, 144, 224, 32, 25)
)

# By default, first column is mapped on the x-axis
# second one on the y axis
billboarder() %>%
  bb_barchart(data = stars)

# Specify explicitly the columns to use
billboarder() %>%
  bb_barchart(data = stars[order(stars$stars), ], x = "package", y = "stars", rotated = TRUE) %>%
  bb_data(names = list(stars = "Number of stars")) %>%
  bb_y_grid(show = TRUE)
```
# Hack stacked barcharts (to color bar)

```r
stars_wide <- data.frame(
  author = c("dreamRs", "davidgohel", "davidgohel", "dreamRs",
             "datastorm-open", "datastorm-open", "AntoineGuillot2"),
  package = c("billboarder", "ggiraph", "officer",
              "shinyWidgets", "visNetwork", "rAmCharts",
              "D3partitionR"),
  stars = c(67, 252, 160, 144, 224, 32, 25)
)

billboarder() %>%
  bb_barchart(data = stars_wide,
              mapping = bbaes(package, stars, group = author),
              stacked = TRUE)

billboarder() %>%
  bb_barchart(data = stars_wide,
              mapping = bbaes(author, stars, group = package),
              stacked = TRUE)
```

# Grouping variable

```r
tab <- table(sample(letters[1:5], 100, TRUE), sample(LETTERS[1:5], 100, TRUE))
dat <- as.data.frame(tab)

billboarder() %>%
  bb_barchart(data = dat, bbaes(x = Var1, y = Freq, group = Var2), rotated = TRUE)
```

# You can also pass data in a 'wide' format

```r
dat2 <- data.frame(
  x = letters[1:5],
  A = sample.int(n = 100, size = 5),
  B = sample.int(n = 100, size = 5),
  C = sample.int(n = 100, size = 5),
  D = sample.int(n = 100, size = 5),
  E = sample.int(n = 100, size = 5)
)

# But cannot use mapping

billboarder() %>%
  bb_barchart(data = dat2, stacked = TRUE) %>%
  bb_data(order = NULL, labels = TRUE)
```

---

### bb_bar_color_manual

Manual color for barchart

---

**Description**

Manual color for barchart
Usage

    bb_bar_color_manual(bb, values)

Arguments

   bb                A billboard htmlwidget object.
   values            A named vector, names represent the categories of the bar chart, values correspond to colors. All categories must be present in the vector, in the same order of the chart.

Value

   A billboard htmlwidget object.

Note

   Must be called after bb_bar.

Examples

   ## Not run:
   library("data.table")
   library("billboarder")
   data("mpg", package = "ggplot2")
   setDT(mpg)
   # all in blue
   manufa <- unique(mpg$manufacturer)
   cols <- rep("#08298A", length(manufa))
   names(cols) <- manufa
   # Nissan in red
   cols["nissan"] <- "#DF0101"
   billboarder() %>%
     bb_barchart(data = mpg[, list(count = .N), by = manufacturer[order(count)]] %>%
                  bb_bar_color_manual(values = cols)
   ## End(Not run)
Usage

bb_bubble(bb, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bb</td>
<td>A billboard htmlwidget object.</td>
</tr>
<tr>
<td>...</td>
<td>See <a href="https://naver.github.io/billboard.js/release/latest/doc/Options.html#bubble">https://naver.github.io/billboard.js/release/latest/doc/Options.html#bubble</a></td>
</tr>
</tbody>
</table>

Value

A billboard htmlwidget object.

Examples

```r
billboarder() `%>%`
  bb_scatterplot(
    data = iris,
    mapping = bbaes(Sepal.Length, Sepal.Width, group = Species, size = Petal.Width)
  ) `%>%`
  bb_bubble(maxR = 10)

billboarder() `%>%`
  bb_scatterplot(
    data = iris,
    mapping = bbaes(Sepal.Length, Sepal.Width, group = Species, size = Petal.Width)
  ) `%>%`
  bb_bubble(maxR = JS("function(d) {return Math.sqrt(d.value.z * 20);}")))```

bb_callbacks

Callbacks for billboard charts

Description

Callbacks for billboard charts

Usage

bb_callbacks(
  bb,
  onafterinit = NULL,
  onbeforeinit = NULL,
  oninit = NULL,
  onout = NULL,
  onover = NULL,
  onrendered = NULL,
  onresize = NULL,
  onresized = NULL
)
**bb_categories**

**Arguments**

- **bb**: A billboard htmlwidget object.
- **onafterinit**: Set a callback to execute after the chart is initialized.
- **onbeforeinit**: Set a callback to execute before the chart is initialized.
- **oninit**: Set a callback to execute when the chart is initialized.
- **onout**: Set a callback to execute when mouse/touch leaves the chart.
- **onover**: Set a callback to execute when mouse/touch enters the chart.
- **onrendered**: Set a callback which is executed when the chart is rendered. Basically, this callback will be called in each time when the chart is redrawed.
- **onresize**: Set a callback to execute when user resizes the screen.
- **onresized**: Set a callback to execute when screen resize finished.

**Value**

A billboard htmlwidget object.

**Note**

Set JavaScript callbacks for various billboard events. See the billboard options reference for additional details on the signature of each callback.

---

**bb_categories**  
*Set categories on X axis*

**Description**

Set or modify x axis labels.

**Usage**

`bb_categories(bb, categories)`

**Arguments**

- **bb**: A billboard htmlwidget object.
- **categories**: A character vector to set names on a category axis.

**Value**

A billboard htmlwidget object.

**Note**

This function can be used with `billboarder-shiny` to modify labels on axis, e.g. for barcharts.
Examples

# Simple line with month names as x labels
billboarder() %>%
  bb_linechart(data = round(rnorm(12))) %>%
  bb_categories(categories = month.name)

bb_color

Color property for a Billboard.js chart

Description

Color property for a Billboard.js chart

Usage

bb_color(bb, palette = NULL, ...)

Arguments

bb
A billboard htmlwidget object.

palette
A color palette to use with series added in the chart.

... See https://naver.github.io/billboard.js/release/latest/doc/Options.html#color

Value

A billboard htmlwidget object.

Examples

library("RColorBrewer")

# Scatter
billboarder() %>%
  bb_scatterplot(data = iris, x = "Sepal.Length", y = "Sepal.Width", group = "Species") %>%
  bb_axis(x = list(tick = list(fit = FALSE))) %>%
  bb_point(r = 8) %>%
  bb_color(palette = brewer.pal(n = 3, name = "Reds"))

# Pie
stars <- data.frame(
  package = c("billboarder", "ggiraph", "officer", "shinyWidgets", "visNetwork"),
  stars = c(9, 177, 43, 44, 169)
)
cols <- brewer.pal(n = 5, name = "Dark2")
bb_colors_manual

bb_colors_manual

Set colors for each data.

Usage

bb_colors_manual(bb, ..., opacity = 1)

Arguments

bb
A billboard htmlwidget object.

... A named list, where names correspond to the data, and values to color associate with it.

opacity Color opacity (for area charts).

Examples

library("RColorBrewer")

# Scatter
billboarder() %>%
  bb_scatterplot(
    data = iris,
    x = "Sepal.Length",
    y = "Sepal.Width",
    group = "Species"
  ) %>%
  bb_axis(x = list(tick = list(fit = FALSE))) %>%
  bb_point(r = 8) %>%
  bb_colors_manual(
    setosa = "#440154",
    virginica = "#21908C",
    versicolor = "#FDE725"
  )

# Pie
stars <- data.frame(
    package = c("billboarder", "ggiraph", "officer",
                "shinyWidgets", "visNetwork"),
    stars = c(9, 177, 43, 44, 169)
) 
cols <- brewer.pal(n = 5, name = "Dark2")

billboarder() %>%
bb_piechart(data = stars) %>%
bb_colors_manual(
    setNames(as.list(cols), stars$package) # this is a named list
)

---

**bb_data**  
*Add data to Billboard chart*

**Description**

Add data to Billboard chart

**Usage**

bb_data(bb, ...)

**Arguments**

- **bb**
  - A billboard htmlwidget object.
- **...**
  - Arguments defined in [https://naver.github.io/billboard.js/demo/](https://naver.github.io/billboard.js/demo/).

**Value**

A billboard htmlwidget object.

**Note**

This function can be used with `billboarderProxy` in shiny application.

**Examples**

```{r}
billboarder() %>%
bb_barchart(data = table(mtcars$cyl)) %>%
bb_data(names = list(Freq = "Number of cylinders"), labels = TRUE)
```
bb_densityplot  

Helper for creating a density plot

Description

Helper for creating a density plot

Usage

bb_densityplot(
  bb,          
  data,        
  mapping = NULL, 
  stacked = FALSE, 
  stat = "density", 
  fill = FALSE, 
  ...          
)

Arguments

- **bb** A billboard htmlwidget object.
- **data** A data.frame or a vector, the first column will be used to calculate density if x is NULL.
- **mapping** Mapping of variables on the chart, see bbaes.
- **stacked** Logical, create a stacked density plot.
- **stat** Stat to compute: density or count.
- **fill** Produce a conditional density estimate, this option force stacked = TRUE.
- **...** Arguments passed to density.

Value

A billboard htmlwidget object.

See Also

bb_histogram

Examples

```r
# With a vector
billboarder() %>%
  bb_densityplot(data = rnorm(1e4))

data("diamonds", package = "ggplot2")
```
# density plot with one variable
billboarder() %>%
  bb_densityplot(data = diamonds, x = "carat")

# Same with mapping
billboarder() %>%
  bb_densityplot(diamonds, bbaes(carat))

# With a grouping variable
billboarder() %>%
  bb_densityplot(data = diamonds, x = "depth", group = "cut") %>%
  bb_x_axis(min = 55, max = 70)

# Same with mapping
billboarder() %>%
  bb_densityplot(diamonds, bbaes(depth, group = cut)) %>%
  bb_x_axis(min = 55, max = 70)

# a stacked density plot using count as statistic
bb <- billboarder() %>%
  bb_densityplot(diamonds, bbaes(depth, group = cut),
                 stacked = TRUE, stat = "count") %>%
  bb_x_axis(min = 55, max = 70)
bb

# changing order
bb %>% bb_data(order = "asc")

---

**bb_donut**  
*Donut property for a Billboard.js chart*

**Description**

Donut property for a Billboard.js chart

**Usage**

```r
bb_donut(bb, ...)
```

**Arguments**

- **bb**  
  A billboard htmlwidget object.

- **...**  
  See https://naver.github.io/billboard.js/release/latest/doc/Options.html#donut
bb_donutchart

Value

A billboard htmlwidget object.

Examples

```r
billboarder() %>%
  bb_donutchart(data = table(mtcars$cyl)) %>%
  bb_donut(title = "Donut Title", width = 10)
```

bb_donutchart                Helper for creating a donut chart

Description

Helper for creating a donut chart

Usage

```r
bb_donutchart(bb, data, mapping = NULL, ...)
```

Arguments

```r
bb
A billboard htmlwidget object.

data
A data.frame.

mapping
Mapping of variables on the chart, see bbaes.

...
```

Value

A billboard htmlwidget object.

Note

This function can be used with billboarderProxy in shiny application.

Examples

```r
## Not run:
stars <- data.frame(
  package = c("billboarder", "ggiraph", "officer", "shinyWidgets", "visNetwork"),
  stars = c(9, 177, 43, 44, 169)
)

billboarder() %>%
  bb_donutchart(data = stars, title = "Stars")

## End(Not run)
```
**bb_export**  

*Export a Billboard to PNG*

**Description**

Export a Billboard to PNG

**Usage**

```
bb_export(bb, filename = NULL, download_label = "Export (.png)", ...)
```

**Arguments**

- **bb**: A `billboarder` htmlwidget object or a `billboarderProxy` htmlwidget object.
- **filename**: A string of the filename, excluding extension (will be ".png").
- **download_label**: Label to appear on the link to download PNG.
- **...**: Additional arguments (not used).

**Value**

A billboard htmlwidget object.

**Note**

This function has two uses:

- **in shiny**: you can export to PNG with an observeEvent by using `billboarderProxy`.
- **in markdown and in shiny**: add a button to download chart as PNG.

**Examples**

```r
# Add a button to download as PNG:

data("equilibre_mensuel")
billboarder() %>%
  bb_linechart(
    data = equilibre_mensuel,
    mapping = bbaes(date, solde),
    type = "spline"
  ) %>%
  bb_x_axis(
    tick = list(format = "%Y-%m", fit = FALSE)
  ) %>%
  bb_export(
    filename = "my-awesome-chart",
    download_label = "Click to download"
  )
```
In shiny, you can use proxy:

```r
if (interactive()) {
  library(shiny)
  library(billboarder)

  ui <- fluidPage(
    fluidRow(
      column(
        width = 8, offset = 2,
        tags$h1("Export billboard as PNG via Proxy"),
        billboarderOutput(outputId = "mybb"),
        actionButton(
          inputId = "export",
          label = "Export",
          icon = icon("download")
        )
      )
    )
  )

  server <- function(input, output, session) {
    output$mybb <- renderBillboarder({
      data("prod_par_filiere")
      billboarder() %>%
        bb_barchart(
          data = prod_par_filiere[, c("annee", "prod_hydraulique")],
          color = "#102246"
        ) %>%
        bb_y_grid(show = TRUE)
    })

    observeEvent(input$export, {
      billboarderProxy(shinyId = "mybb") %>%
        bb_export(filename = "my-billboard-chart")
    })
  }

  shinyApp(ui, server)
}
```

---

**bb_gauge**

Gauge property for a Billboard.js chart

---

**Description**

Gauge property for a Billboard.js chart
Usage

bb_gauge(bb, ...)

Arguments

bb

A billboard htmlwidget object.

... 

See https://naver.github.io/billboard.js/release/latest/doc/Options.html#.gauge

Value

A billboard htmlwidget object.

Examples

billboarder() %>%
  bb_gaugechart(value = 50) %>%
  bb_gauge(min = 0, max = 200, units = "km/h", width = 10,
            label = list(format = htmlwidgets::JS("function(value) {return value;}")))

bb_gaugechart

Helper for creating a gauge

Description

Helper for creating a gauge

Usage

bb_gaugechart(
  bb,
  value,
  name = "Value",
  steps = c(30, 60, 90, 100),
  steps_color = c("#FF0000", "#F97600", "#F6C600", "#60B044"),
  ... 
)

Arguments

bb

A billboard htmlwidget object.

value

A numeric value.

name

Name for the value, appear in tooltip.

steps

Upper bound for changing colors

steps_color

Colors corresponding to steps

... 

Arguments for slot gauge.
bb_grid

Value

A billboard htmlwidget object.

Note

This function can be used with billboarderProxy in shiny application.

Examples

```r
billboader() %>%
    bb_gaugechart(value = 50)

# With some options
billboader() %>%
    bb_gaugechart(
        value = 160,
        steps_color = rev(c("#FF0000", "#F97600", "#F6C600", "#60B044"))
    ) %>%
    bb_gauge(
        label = list(format = suffix("km/h")),
        min = 10, max = 200, width = 20
    )
```

bb_grid

Grid property for a Billboard.js chart

Description

Grid property for a Billboard.js chart

Usage

```r
bb_grid(bb, ...)
bb_x_grid(bb, ...)
bb_y_grid(bb, ...)
```

Arguments

- `bb`: A billboard htmlwidget object.
- `...`: See [https://naver.github.io/billboard.js/release/latest/doc/Options.html#grid](https://naver.github.io/billboard.js/release/latest/doc/Options.html#grid)

Value

A billboard htmlwidget object.
**Note**

`bb_x_grid` and `bb_y_grid` are shortcut for modifying the x-axis and the y-axis respectively.

**Examples**

```r
stars <- data.frame(
  package = c("billboarder", "ggiraph", "officer", "shinyWidgets", "visNetwork"),
  stars = c(1, 176, 42, 40, 166)
)

billboarder() %>%
  bb_barchart(data = stars) %>%
  bb_y_grid(show = TRUE)

billboarder() %>%
  bb_barchart(data = stars) %>%
  bb_y_grid(lines = list(list(value = mean(stars$stars), text = "Horizontal line")))
```

---

**bb_histogram**

*Helper for creating an histogram*

**Description**

Helper for creating an histogram

**Usage**

```r
bb_histogram(
  bb, 
  data, 
  mapping = NULL, 
  stacked = FALSE, 
  fill = FALSE, 
  bins = 30, 
  binwidth = NULL, 
  ...
)
```

**Arguments**

- `bb`: A `billboard htmlwidget` object.
- `data`: A `data.frame` or a `vector`, the first column will be used to calculate density if `x` is `NULL`.
- `mapping`: Mapping of variables on the chart, see `bbaes`.
- `stacked`: Logical, create a stacked histogram.
fill Logical, create a stacked percentage histogram.
bins Number of bins. Overridden by binwidth. Defaults to 30.
binwidth The width of the bins. See `geom_histogram`

Value
A billboard htmlwidget object.

See Also
`bb_densityplot`

Examples

```r
data("diamonds", package = "ggplot2")

# one variable
billboarder() %>%
  bb_histogram(data = diamonds, x = "price")

# with mapping
billboarder() %>%
  bb_histogram(diamonds, bbaes(price))

# equivalent to
billboarder() %>%
  bb_histogram(data = diamonds$price)

# prettier with 'binwidth'
# (but you need to know your data)
billboarder() %>%
  bb_histogram(data = diamonds, x = "price", binwidth = 500) %>%
  bb_colors_manual()

# with a grouping variable
billboarder() %>%
  bb_histogram(data = diamonds, x = "price",
               group = "cut", binwidth = 500)

# and with mapping
billboarder() %>%
  bb_histogram(diamonds, bbaes(price, group = cut),
               binwidth = 500)

# stacked histogram
billboarder() %>%
  bb_histogram(diamonds, bbaes(price, group = cut),
               stacked = TRUE, binwidth = 500)
```
# another example

dat <- data.frame(
    sample = c(rnorm(n = 500, mean = 1), rnorm(n = 500, mean = 2)),
    group = rep(c("A", "B"), each = 500)
)

billboarder() %>%
  bb_histogram(data = dat, x = "sample", binwidth = 0.25)

samples_mean <- tapply(dat$sample, dat$group, mean)

billboarder() %>%
  bb_histogram(data = dat, x = "sample", group = "group",
               binwidth = 0.25) %>%
  bb_x_grid(
    lines = list(
      list(value = unname(samples_mean["A"]),
             text = "mean of sample A"),
      list(value = unname(samples_mean["B"]),
             text = "mean of sample B")
    )
)

---

**bb_interaction**  
*Interaction property for a Billboard.js chart*

**Description**

Interaction property for a Billboard.js chart

**Usage**

bb_interaction(bb, ...)

**Arguments**

- **bb**  
  A billboard htmlwidget object.

- **...**  
  See [https://naver.github.io/billboard.js/release/latest/doc/Options.html#interaction](https://naver.github.io/billboard.js/release/latest/doc/Options.html#interaction)

**Value**

A billboard htmlwidget object.
Description

Quickly set title, axis labels and caption

Usage

```r
bb_labs(
    bb,
    title = NULL,
    x = NULL,
    y = NULL,
    caption = NULL,
    caption_href = NULL
)
```

Arguments

- `bb`: A billboard htmlwidget object.
- `title`: Text for the chart title, use \n to make a new line.
- `x`: Text for x axis title.
- `y`: Text for y axis title.
- `caption`: Text for the caption displayed in the bottom-right of the chart.
- `caption_href`: Associate the caption with a link to an URL.

Value

A billboard htmlwidget object.

Note

caption is not part of the billboard.js library, it is added by the billieder package.

Examples

```r
data("prod_par_filiere")

billieder() %>%
  bb_barchart(
    data = prod_par_filiere[, c("annee", "prod_hydraulique")],
    color = "#102246"
  ) %>%
  bb_legend(show = FALSE) %>%
  bb_labs(
    title = "French hydraulic production",
```
bb_legend

Add legend parameters

Description
Add legend parameters

Usage
bb_legend(bb, ...)

Arguments

bb  A billboard htmlwidget object.
...

Value
A billboard htmlwidget object.

Examples

library("billboarder")

stars <- data.frame(
  package = c("billboarder", "ggtree", "officer", "shinyWidgets", "visNetwork"),
  stars = c(1, 176, 42, 40, 166)
)

# Hide legend
billboarder() %>%
  bb_barchart(data = stars) %>%
  bb_legend(show = FALSE)

# Right legend
billboarder() %>%
  bb_piechart(data = stars) %>%
  bb_legend(position = "right")

# Inset legend
billboarder() %>%
  bb_scatterplot(data = iris, x = "Sepal.Length", y = "Sepal.Width", group = "Species") %>%
  bb_axis(x = list(tick = list(fit = FALSE))) %>%
  bb_legend(position = "inset", inset = list(anch = "top-right"))
**bb_line**

*Line property for a Billboard.js chart*

**Description**

Line property for a Billboard.js chart

**Usage**

`bb_line(bb, ...)`

**Arguments**

- `bb` A billboard htmlwidget object.
- `...` See [https://naver.github.io/billboard.js/release/latest/doc/Options.html#].

**Value**

A billboard htmlwidget object.

**Examples**

```r
# Set if null data point will be connected or not.
b <- billboarder() %>%
  bb_linechart(data = c(1, 2, NA, 4, 5))
b %>% bb_line(connectNull = TRUE)
```

---

**bb_linechart**

*Helper for creating a line chart*

**Description**

Helper for creating a line chart

**Usage**

```r
bb_linechart(
  bb,
  data,
  mapping = NULL,
  type = "line",
  show_point = FALSE,
  dasharray = NULL,
)```
Arguments

bb
A billboard htmlwidget object.

data
A data.frame or a vector.

mapping
Mapping of variables on the chart, see bbaes.

type
Type of chart: "line", "spline", "step", "area", "area-spline", "area-step", "area-line-range", "area-spline-range".

show_point
Whether to show each point in line.

dasharray
Pattern of dashes and gaps used to paint the outline of the line, see https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-dasharray for specifications.

width
Width of the stroke to be applied to the line, see https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-width for specifications.

Value

A billboard htmlwidget object.

Note

Types area-line-range and area-spline-range don’t work in RStudio viewer, open chart in a browser. This function can be used with billboarderProxy in shiny application.

Examples

```r
## Different types
x <- round(rnorm(20), 2)

billboarder() %>%
  bb_linechart(data = x)

billboarder() %>%
  bb_linechart(data = x, type = "spline")

billboarder() %>%
  bb_linechart(data = x, type = "area")

billboarder() %>%
  bb_linechart(data = x, type = "area-spline")

## Timeserie with date (Date)
data("economics", package = "ggplot2")
```
bb_linechart

billboarder() %>%
  bb_linechart(data = economics[, c("date", "psavert")]) %>%
  bb_x_axis(tick = list(format = "%Y-%m", fit = FALSE)) %>%
  bb_y_axis(tick = list(format = suffix("%")),
             label = list(text = "Personal savings rate")) %>%
  bb_legend(show = FALSE) %>%
  bb_x_grid(show = TRUE) %>%
  bb_y_grid(show = TRUE) %>%
  bb_subchart(show = TRUE)

# With multiple lines :

data("economics_long", package = "ggplot2")

billboarder() %>%
  bb_linechart(economics_long, bbaes(date, value, variable)) %>%
  bb_data(hide = "pop") %>%
  bb_x_axis(tick = list(format = "%Y-%m", fit = FALSE))

## Timeserie with datetime (POSIXct)
data("cdc_prod_filiere")

billboarder() %>%
  bb_linechart(data = cdc_prod_filiere[, c("date_heure", "prod_eolien")])

# or with mapping :
billboarder() %>%
  bb_linechart(cdc_prod_filiere, bbaes(date_heure, prod_bioenergies))

### Other type for x-axis

### character/factor on x-axis
AirPassengers1960 <- data.frame(
  month = month.name,
  AirPassengers = tail(AirPassengers, 12)
)

# you have to specify that x-axis is of type 'category'
# and that column 'month' must be used for x-axis values
billboarder() %>%
  bb_linechart(data = AirPassengers1960, x = "month") %>%
  bb_x_axis(type = "category")

### numeric on x-axis
lynx.df <- data.frame(
  year = time(lynx),
  lynx = lynx

```r
### Area range charts

# Generate data
dat <- data.frame(
  date = seq.Date(Sys.Date(), length.out = 20, by = "day"),
  y1 = round(rnorm(20, 100, 15)),
  y2 = round(rnorm(20, 100, 15))
)
dat$ymin1 <- dat$y1 - 5
dat$ymax1 <- dat$y1 + 5
dat$ymin2 <- dat$y2 - sample(3:15, 20, TRUE)
dat$ymax2 <- dat$y2 + sample(3:15, 20, TRUE)

# Make chart: use ymin & ymax aes for range
billboarder(data = dat) %>%
  bb_linechart(
    mapping = bbaes(x = date, y = y1, ymin = ymin1, ymax = ymax1),
    type = "area-line-range"
  ) %>%
  bb_linechart(
    mapping = bbaes(x = date, y = y2, ymin = ymin2, ymax = ymax2),
    type = "area-spline-range"
  ) %>%
  bb_y_axis(min = 50)
```

---

### bb_load

*Load data to the chart with proxy*

**Description**

Load data to the chart with proxy

**Usage**

```
bb_load(proxy, data = NULL, unload = NULL, ...)
```

**Arguments**

- `proxy` A billboardProxy htmlwidget object.
- `data` A data.frame with updated data.
- `unload` Ids (names) to data to unload.
- `...` Arguments passed to method.
**bb_lollipop**

**Value**

A `billboardProxy htmlwidget` object.

**Description**

Helper for creating a lollipop chart

**Usage**

```r
bb_lollipop(
  bb,
  data,
  mapping = NULL,
  rotated = FALSE,
  point_color = "#112446",
  point_size = 8,
  line_color = "#000",
  ...
)
```

**Arguments**

- **bb**
  A `billboard htmlwidget` object.
- **data**
  A `data.frame`, the first column will be used for x axis unless argument x is specified, the second one will be use as y values. If not a `data.frame`, an object coercible to `data.frame`.
- **mapping**
  Mapping of variables on the chart, see `bbaes`.
- **rotated**
  Switch x and y axis position.
- **point_color**
  Color of the lollipop.
- **point_size**
  Size of the lollipop.
- **line_color**
  Color of the lines between the axis and the lollipop.
- **...**
  Not used.

**Value**

A `billboard htmlwidget` object.
```r
# From wikipedia
sw <- data.frame(
  film = c("The Force Awakens", "The Phantom Menace",
            "Revenge of the Sith", "A New Hope",
            "Attack of the Clones", "The Empire Strikes Back",
            "Return of the Jedi" ),
  worldwide_gross = c(2068178225, 1027044677, 848754768,
                      775398007, 649398328, 538375067,
                      475106177))

# Simple example
billboarder() %>%
  bb_lollipop(data = sw)

# Fancy example
billboarder() %>%
  bb_lollipop(data = sw, rotated = TRUE) %>%
  bb_y_grid(show = TRUE) %>%
  bb_y_axis(tick = list(
    values = c(0, 5e+08, 1e+09, 1.5e+09, 2e+09),
    outer = FALSE,
    format = htmlwidgets::JS("d3.formatPrefix('$.0', 1e6)")) %>%
  bb_x_axis(tick = list(centered = TRUE)) %>%
  bb_labs(
    title = "Star Wars - Total Lifetime Grosses",
    caption = "Data source : wikipedia"
  )

# With mapping
billboarder(data = sw) %>%
  bb_lollipop(mapping = bbaes(x = film, y = worldwide_gross))
```

---

**bb_pie**  
*Pie property for a Billboard.js chart*

**Description**  
Pie property for a Billboard.js chart

**Usage**  
`bb_pie(bb, ...)`
Arguments

bb  A billboard htmlwidget object.
...
See https://naver.github.io/billboard.js/release/latest/doc/Options.html#.pie

Value

A billboard htmlwidget object.

Examples

billboarder() %>%
  bb_piechart(data = table(mtcars$cyl)) %>%
  bb_pie(label = list(
    ratio = 0.5,
    format = htmlwidgets::JS("function(value) {return d3.format(''$'(value);}())
  ),
  expand = FALSE)

bb_piechart  Helper for creating a pie chart

Description

Helper for creating a pie chart

Usage

bb_piechart(bb, data, mapping = NULL, ...)

Arguments

bb  A billboard htmlwidget object.
data  A data.frame, first column should contain labels, second column values associated, except if mapping is provided.
mapping  Mapping of variables on the chart, see bbaes.
...

Value

A billboard htmlwidget object.

Note

This function can be used with billboarderProxy in shiny application.
Examples

stars <- data.frame(
  package = c("billboarder", "ggiraph", "officer", "shinyWidgets", "visNetwork"),
  stars = c(9, 177, 43, 44, 169)
)

# Default
billboarder() %>%
  bb_piechart(data = stars)

# Explicit mapping
billboarder() %>%
  bb_piechart(data = stars, bbaes(package, stars))

# Other way to specify mapping
billboarder(data = stars) %>%
  bb_aes(package, stars) %>%
  bb_piechart()

---

bb_point  

Point property for a Billboard.js chart

Description

Point property for a Billboard.js chart

Usage

bb_point(bb, ...)

Arguments

bb  
  A billboard htmlwidget object.
...
  See [https://naver.github.io/billboard.js/release/latest/doc/Options.html#point](https://naver.github.io/billboard.js/release/latest/doc/Options.html#point)

Value

A billboard htmlwidget object.

Examples

# Set point size
billboarder() %>%
  bb_scatterplot(data = iris, x = "Sepal.Length", y = "Sepal.Width", group = "Species") %>%
  bb_axis(x = list(tick = list(fit = FALSE))) %>%
  bb_point(r = 10)
bb_proxy_axis_labels  Update axis labels with proxy

**Description**
Update axis labels with proxy

**Usage**
```r
bb_proxy_axis_labels(proxy, x = NULL, y = NULL)
```

**Arguments**
- `proxy`  A billboardProxy htmlwidget object.
- `x`  X axis label.
- `y`  Y axis label.

**Value**
A billboardProxy htmlwidget object.

---

bb_proxy_data_colors  Change colors with proxy

**Description**
Change colors with proxy

**Usage**
```r
bb_proxy_data_colors(proxy, names = NULL, colors = NULL)
```

**Arguments**
- `proxy`  A billboardProxy htmlwidget object.
- `names`  Names of series
- `colors`  New colors, in same order that names.

**Value**
A billboardProxy htmlwidget object.
Examples

```r
if (interactive()) {
  library(shiny)
  library(billboarder)

  ui <- fluidPage(
    tags$h2("Update colors"),
    fluidRow(
      column(
        width = 3,
        selectizeInput(
          inputId = "col_eol",
          label = "Color for 'prod_eolien':",
          choices = c("#66C2A5", "#FC8D62",
                       "#8DA0CB", "#E78AC3",
                       "#A6D854", "#FFD92F",
                       "#E5C494", "#B3B3B3")
        ),
        selectizeInput(
          inputId = "col_sol",
          label = "Color for 'prod_solaire':",
          choices = c("#66C2A5", "#FC8D62",
                       "#8DA0CB", "#E78AC3",
                       "#A6D854", "#FFD92F",
                       "#E5C494", "#B3B3B3")
        ),
        column(
          width = 9,
          billboarderOutput(outputId = "my_bb")
        )
      )
    )
  )

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

    output$my_bb <- renderBillboarder({
      data(prod_par_filiere)
      billboarder() %>%
      bb_barchart(
        data = prod_par_filiere[, c(1, 6, 8)]
      )
    })

    observe({
      billboarderProxy(shinyId = "my_bb") %>%
      bb_proxy_data_colors(
        names = c("prod_eolien", "prod_solaire"),
        colors = c(input$col_eol, input$col_sol)
      )
    })
  }
}
```
bb_proxy_data_names

}  
shinyApp(ui, server)  
}

bb_proxy_data_names  Change names of the data with proxy

Description
Change names of the data with proxy

Usage
bb_proxy_data_names(proxy, old = NULL, new = NULL)

Arguments
proxy  A billboardProxy htmlwidget object.
old  Old names
new  New names

Value
A billboardProxy htmlwidget object.

Examples
if (interactive()) {

library(shiny)
library(billboarder)

ui <- fluidPage(
  tags$h2("Update axis title & data name (tooltip & legend)"),
  billboarderOutput(outputId = "my_bb"),
  textInput(
    inputId = "new_name",
    label = "New name:",
    value = "this is a new name",
    width = "100%"
  ),
  actionButton(
    inputId = "update",
    label = "Update chart",
    width = "100%"
  )
)
server <- function(input, output, session) {

  output$my_bb <- renderBillboarder({
    dat <- sample(letters[1:5], 100, TRUE)
    billboarder() %>%
      bb_barchart(data = table(dat)) %>%
      bb_y_axis(label = list(text = "Freq"))
  })

  observeEvent(input$update, {
    dat <- sample(letters[1:5], 100, TRUE)
    billboarderProxy(shinyId = "my_bb") %>%
      bb_proxy_axis_labels(y = input$new_name) %>%
      bb_proxy_data_names(old = "Freq",
                         new = input$new_name) %>%
    bb_barchart(data = table(dat)))
  }, ignoreInit = TRUE)
}

shinyApp(ui, server)

---

**bb_proxy_flow**  
*Update chart flow with proxy*

**Description**

Update chart flow with proxy

**Usage**

```
bb_proxy_flow(proxy, ...)
```

**Arguments**

- `proxy` A billboardProxy htmlwidget object.
- `...` Arguments passed to the flow API, see [https://naver.github.io/billboard.js/release/latest/doc/Chart.html#flow](https://naver.github.io/billboard.js/release/latest/doc/Chart.html#flow).

**Value**

A billboardProxy htmlwidget object.
Examples

```r
if (interactive()) {
  library(shiny)
  library(billboarder)

  ui <- fluidPage(
    tags$h3("Proxy flow"),
    actionButton(
      inputId = "next_data",
      label = "Add data",
      icon = icon("arrow-right")
    ),
    billboarderOutput(outputId = "chart1"),
    tags$h4("Real time chart"),
    billboarderOutput(outputId = "chart2")
  )

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

    time_data <- reactiveValues(df = data.frame(
      x = Sys.Date() + 1:20,
      y = round(rnorm(20) * 10)
    ))

    output$chart1 <- renderBillboarder({
      billboarder() %>%
        bb_linechart(data = isolate(time_data$df))
    })

    observeEvent(input$next_data, {
      time_data$df$x <- time_data$df$x + 21
      time_data$df$y <- round(rnorm(20) * 10)
    })

    observe({
      billboarderProxy("chart1") %>%
        bb_proxy_flow(json = as.list(time_data$df), duration = 1500)
    })

    output$chart2 <- renderBillboarder({
      df <- data.frame(
        x = Sys.time() - 1:20 * 2,
        y = round(rnorm(20) * 10)
      )
      billboarder() %>%
        bb_linechart(data = df) %>%
        bb_x_axis(tick = list(format = "%H:%M", fit = FALSE))
    })
  }
}
```
observe({
  invalidateLater(2000)
  billboarderProxy("chart2") %>%
    bb_proxy_flow(json = list(
      x = list(format(Sys.time())),
      y = list(round(rnorm(1) * 10))
    ), data = list(x = "x"))
})

shinyApp(ui, server)

bb_proxy_focus

Highlights specified targets and fade out the others.

Description
Highlights specified targets and fade out the others.

Usage
bb_proxy_focus(proxy, ids = NULL)
bb_proxy_defocus(proxy, ids = NULL)

Arguments
proxy A billboardProxy htmlwidget object.
ids Data ids (names) to be highlighted, if NULL all datas will be highlighted.

Value
A billboardProxy htmlwidget object.

Note
bb_defocus is the opposite of bb_focus

Examples
if (interactive()) {
  library("shiny")
  library("billboarder")

  ui <- fluidPage(
    tags$h1("Proxy method to highlight data"),
  )
```r
checkboxGroupInput(
  inputId = "focus",
  label = "Focus",
  choices = c("setosa", "versicolor", "virginica"),
  inline = TRUE
),
billboarderOutput(outputId = "bb")
}

server <- function(input, output, session) {
  output$bb <- renderBillboarder({
    billboarder() %>%
      bb_scatter(
        data = iris,
        x = "Sepal.Length",
        y = "Sepal.Width",
        group = "Species"
      ) %>%
      bb_axis(x = list(tick = list(fit = FALSE))) %>%
      bb_point(r = 8)
  })

  observeEvent(input$focus, {
    billboarderProxy("bb") %>%
      bb_proxy_focus(input$focus)
  }, ignoreNULL = FALSE)
}

shinyApp(ui = ui, server = server)
```

### bb_proxy_groups

**Update chart groups with proxy**

**Description**

Update chart groups with proxy

**Usage**

```r
bb_proxy_groups(proxy, ...)
```

**Arguments**

- `proxy` A `billboardProxy` htmlwidget object.
- `...` Vector(s) with id of the series, e.g. the name of variables.

**Value**

A `billboardProxy` htmlwidget object.
**bb_proxy_hide**  
*Hide method with proxy*

**Description**  
Hide method with proxy

**Usage**  
```r
bb_proxy_hide(proxy, targetIdsValue, options = NULL)
```

**Arguments**

- `proxy`  
  A `billboardProxy htmlwidget` object.

- `targetIdsValue`  
  Name of series to hide.

- `options`  
  Additional options.

**Value**  
A `billboardProxy htmlwidget` object.

**See Also**

- `bb_proxy_show`

---

**bb_proxy_legend**  
*Show or hide legend with proxy*

**Description**  
Show or hide legend with proxy

**Usage**  
```r
bb_proxy_legend(proxy, what = c("show", "hide"), targetIds = NULL)
```

**Arguments**

- `proxy`  
  A `billboardProxy htmlwidget` object.

- `what`  
  Show or hide the legend.

- `targetIds`  
  Series ids to show/hide, if NULL show/hide all legend.

**Value**  
A `billboardProxy htmlwidget` object.
Examples

```r
if (interactive()) {
    library("shiny")
    data("prod_par_filiere")

    ui <- fluidPage(
        tags$h2("Show or hide legend with Proxy"),
        fluidRow(
            column(
                width = 3,
                checkboxInput(
                    inputId = "show_legend",
                    label = "Show legend",
                    value = TRUE
                ),
                checkboxGroupInput(
                    inputId = "item_show",
                    label = "Item to show in legend",
                    choices = c("Hydraulic" = "prod_hydraulique",
                                "Wind" = "prod_eolien",
                                "Solar" = "prod_solaire"),
                    selected = c("prod_hydraulique", "prod_eolien", "prod_solaire")
                )
            ),
            column(
                width = 9,
                billboarderOutput(outputId = "mybb")
            )
        )
    )

    server <- function(input, output, session) {
        output$mybb <- renderBillboarder({
            billboarder() %>%
            bb_barchart(
                data = prod_par_filiere[, c("annee", "prod_hydraulique", "prod_eolien", "prod_solaire")],
                stacked = TRUE
            ) %>%
            bb_data(
                names = list(prod_hydraulique = "Hydraulic",
                             prod_eolien = "Wind",
                             prod_solaire = "Solar"),
                labels = TRUE
            ) %>%
            bb_colors_manual(
                "prod_eolien" = "#41AB5D",
                "prod_hydraulique" = "#4292C6",
                "prod_solaire" = "#3F51B5"
            )
        } )
    }
}
```
"prod_solaire" = "#FEB24C"
) %>%
bb_y_grid(show = TRUE) %>%
bb_y_axis(
  tick = list(format = suffix("TWh")),
  label = list(text = "production (in terawatt-hours)",
               position = "outer-top")
) %>%
bb_legend(position = "right") %>%
bb_labs(
  title = "Renewable energy production",
  caption = "Data source: RTE (https://opendata.rte-france.com)"
)
})

observe(
  if (input$show_legend) {
    billboarderProxy("mybb") %>%
    bb_proxy_legend(what = "show")
  } else {
    billboarderProxy("mybb") %>%
    bb_proxy_legend(what = "hide")
  }
)

observe(
  lapply(
    X = c("prod_hydraulique", "prod_eolien", "prod_solaire"),
    FUN = function(x) {
      if (x %in% input$item_show) {
        billboarderProxy("mybb") %>%
        bb_proxy_legend(what = "show", targetIds = x)
      } else {
        billboarderProxy("mybb") %>%
        bb_proxy_legend(what = "hide", targetIds = x)
      }
    }
  )
)
)

shinyApp(ui = ui, server = server)

---

**bb_proxy_show**

**Show method with proxy**

**Description**

Show method with proxy
Usage

bb_proxy_show(proxy, targetIdsValue, options = NULL)

Arguments

proxy: A billboardProxy htmlwidget object.
targetIdsValue: Name of series to show.
options: Additional options.

Value

A billboardProxy htmlwidget object.

See Also

bb_proxy_hide

bb_proxy_tooltip

Show or hide tooltip with proxy

Description

Show or hide tooltip with proxy

Usage

bb_proxy_tooltip(proxy, what = c("show", "hide"), x = NULL, index = NULL, ...)

Arguments

proxy: A billboardProxy htmlwidget object.
what: show or hide the legend.
x: x value on which the tooltip must appear.
index: Index on the x-axis on which the tooltip must appear.
...: Additional arguments passed to method.

Value

A billboardProxy htmlwidget object.
**bb_proxy_transform**  
*Update chart type with proxy*

**Description**  
Update chart type with proxy

**Usage**  
```r
bb_proxy_transform(proxy, type, targetIds = NULL)
```

**Arguments**
- **proxy**: A `billboardProxy htmlwidget` object.
- **type**: Specify the type to be transformed.
- **targetIds**: Specify targets to be transformed. If not given, all targets will be the candidate.

**Value**  
A `billboardProxy htmlwidget` object.

---

**bb_proxy_xs**  
*Update x values with proxy*

**Description**  
Update x values with proxy

**Usage**  
```r
bb_proxy_xs(proxy, xs)
```

**Arguments**
- **proxy**: A `billboardProxy htmlwidget` object.
- **xs**: Named list of vector(s) used for x values.

**Value**  
A `billboardProxy htmlwidget` object.
bb_radar

Radar property for a Billboard.js chart

Description

Radar property for a Billboard.js chart

Usage

bb_radar(bb, ...)

Arguments

bb  A billboard htmlwidget object.
...

See https://naver.github.io/billboard.js/release/latest/doc/Options.html#.radar

Value

A billboard htmlwidget object.

Examples

library("billboarder")
data("avengers")

# number of levels
billboarder() %>%
  bb_radarchart(
    data = avengers,
    mapping = bbaes(x = axis, y = value, group = group)
  ) %>%
  bb_radar(level = list(depth = 4))

# hide levels
billboarder() %>%
  bb_radarchart(
    data = avengers,
    mapping = bbaes(x = axis, y = value, group = group)
  ) %>%
  bb_radar(level = list(show = FALSE))

# max value on axis
billboarder() %>%
  bb_radarchart(
    data = avengers,
    mapping = bbaes(x = axis, y = value, group = group)
  ) %>%
  bb_radar(axis = list(max = 10))
bb_radarchart  

**Helper for creating a radar chart**

**Description**

Helper for creating a radar chart

**Usage**

```r
bb_radarchart(bb, data, mapping = NULL, ...)
```

**Arguments**

- `bb`: A billboard htmlwidget object.
- `data`: A data.frame, the first column will be used for x axis unless specified otherwise in mapping. If not a data.frame, an object coercible to data.frame.
- `mapping`: Mapping of variables on the chart, see `bbaes`.
- `...`: Arguments passed to `bb_radar`.

**Value**

A billboard htmlwidget object.

**Examples**

```r
library("billboarder")

# data about Avengers
data("avengers_wide")

# if not specified, first column is used as x-axis,
# all others are used on y-axis
billboarder() %>%
  bb_radarchart(data = avengers_wide)

# specify explicitly which column to use with mapping
billboarder() %>%
  bb_radarchart(
    data = avengers_wide,
    mapping = bbaes(x = axis, y = 'Captain America')
  )

# with data in "long" format you can use "group" aesthetics
data("avengers")
billboarder() %>%
  bb_radarchart(
    data = avengers,
  )
```
bb_regions

regions property for a Billboard.js chart

Description

Add a shading effect to the background of the chart, to highlight a period for example.

Usage

bb_regions(bb, ...)

Arguments

bb A billboard htmlwidget object.
...

See https://naver.github.io/billboard.js/release/latest/doc/Options.html#.regions

Value

A billboard htmlwidget object.

Note

This function can be used with billboarderProxy in shiny application.

See Also

bb_add_style

Examples

# With a categorical X-axis
dat <- data.frame(
  month = month.abb,
  AirPassengers = tail(AirPassengers, 12)
)
# Highlight Jun/Jul/Aug
billboarder() %>%
  bb_linechart(data = dat, x = "month") %>%
  bb_x_axis(type = "category") %>%
  bb_regions(
    list(start = 4.5, end = 7.5) #' jan = 0
  )

# With a barchart
billboarder() %>%
bb_barchart(data = dat) %>%
bb_regions(
  list(start = 1.5, end = 2.5, class = "custom"),
  list(start = 8, end = 10, class = "foo")
) %>%
bb_add_style(region = list(custom = "fill: red;", foo = "fill: #009246;"))

# With Date X-axis
library("stats")
dat <- data.frame(
  date = seq.Date(from = Sys.Date(), by = "day", length.out = 365),
  var = density(rexp(n = 1000), n = 365)$y
)

billboarder() %>%
bb_linechart(data = dat) %>%
bb_x_axis(tick = list(fit = FALSE)) %>%
bb_y_axis(min = 0, padding = 0) %>%
bb_regions(
  list(start = format(Sys.Date() + 30), end = format(Sys.Date() + 120))
)

# With POSIXct X-axis
dat <- data.frame(
  time = seq.POSIXt(from = Sys.time(), by = "min", length.out = 60),
  var = round(sort(rnorm(60)), 2)
)

billboarder() %>%
bb_linechart(data = dat) %>%
bb_x_axis(tick = list(format = "%H:%M", fit = FALSE)) %>%
bb_regions(
  list(start = format(dat$time[15]),
       end = format(dat$time[30]))
)

---

**bb_scatterplot**

*Helper for creating a scatter chart*

**Description**

Helper for creating a scatter chart
**Usage**

```r
bb_scatterplot(bb, data, mapping = NULL, ..., point_opacity = NULL)
```

**Arguments**

- `bb`: A billboard htmlwidget object.
- `data`: A data.frame
- `mapping`: Mapping of variables on the chart, see `bbaes`.
- `...`: Alternative mapping, you can specify `x = "Sepal.Length"` for example.
- `point_opacity`: Opacity for points, value between `[0, 1]`.

**Value**

A billboard htmlwidget object.

**Note**

This function can be used with `billboarderProxy` in shiny application.

**Examples**

```r
# Use first and second variable by default
billboarder() %>%
  bb_scatterplot(data = iris)

# Explicit mapping
billboarder() %>%
  bb_scatterplot(
    data = iris,
    mapping = bbaes(Petal.Length, Petal.Width)
  ) %>%
  bb_x_axis(tick = list(fit = FALSE))

# Grouping variable
billboarder() %>%
  bb_scatterplot(
    data = iris,
    mapping = bbaes(Sepal.Length, Sepal.Width, group = Species)
  )

# Size variable
billboarder() %>%
  bb_scatterplot(
    data = iris,
    mapping = bbaes(
      Sepal.Length, Sepal.Width,
      group = Species, size = Petal.Width
    )
  )
```
bb_spline  

*Spline property for a Billboard.js chart*

**Description**

Spline property for a Billboard.js chart

**Usage**

```r
bb_spline(bb, ...)
```

**Arguments**

- `bb`: A billboard htmlwidget object.

**Value**

A billboard htmlwidget object.

---

bb_subchart  

*Subchart property for a Billboard.js chart*

**Description**

Create a subchart allowing to zoom and navigate on the chart.

**Usage**

```r
bb_subchart(bb, ...)
```

**Arguments**

- `bb`: A billboard htmlwidget object.

**Value**

A billboard htmlwidget object.
Examples

```r
data("equilibre_mensuel")

billboarder() %>%
  bb_linechart(data = equilibre_mensuel[, c("date", "production")], type = "spline") %>%
  bb_subchart(show = TRUE)
```
Arguments

bb     A billboard htmlwidget object.
text   The chart title.
padding A named list with top, right, bottom, left values.
position A string specifying the position of the title.
...    Additional arguments.

Value

A billboard htmlwidget object.

See Also

bb_labs

Examples

billboarder() %>%
  bb_barchart(data = table(sample(letters, 100, TRUE))) %>%
  bb_title(text = "Random letters", position = "center")

bb_tooltip

Tooltip property for a Billboard.js chart

Description

Tooltip property for a Billboard.js chart

Usage

bb_tooltip(bb, ...)

Arguments

bb     A billboard htmlwidget object.
...    See https://naver.github.io/billboard.js/release/latest/doc/Options.
        html#tooltip

Value

A billboard htmlwidget object.
Examples

```r
# Format tooltip
billboarder() %>%
  bb_scatterplot(data = iris, x = "Sepal.Length", y = "Sepal.Width", group = "Species") %>%
  bb_tooltip(
    format = list(
      # skip the title in tooltip
      title = htmlwidgets::JS("function() {return undefined;}"),
      name = htmlwidgets::JS("function(name, ratio, id, index) {return '';}"),
      value = htmlwidgets::JS("function(value, ratio, id, index) {return id;}"
    )
  )
```

bb_transition  Transition property for a Billboard.js chart

Description

Transition property for a Billboard.js chart

Usage

```r
bb_transition(bb, ...)
```

Arguments

- `bb`: A billboard htmlwidget object.
- `...`: See [https://naver.github.io/billboard.js/release/latest/doc/Options.html#transition](https://naver.github.io/billboard.js/release/latest/doc/Options.html#transition)

Value

A billboard htmlwidget object.

bb_unload  Unload data to the chart with proxy

Description

Unload data to the chart with proxy

Usage

```r
bb_unload(proxy, ids = NULL)
```
Arguments

proxy A billboardProxy htmlwidget object.
ids Data ids to unload.

Value

A billboardProxy htmlwidget object.

---

**bb_zoom**

*Zoom property for a Billboard.js chart*

Description

Zoom property for a Billboard.js chart

Usage

`bb_zoom(bb, ...)`

Arguments

`bb` A billboard htmlwidget object.
`...` See [https://naver.github.io/billboard.js/release/latest/doc/Options.html](https://naver.github.io/billboard.js/release/latest/doc/Options.html#zoom)

Value

A billboard htmlwidget object.

Examples

```r
# data
data("equilibre_mensuel")

# line chart
billboarder() %>%  
  bb_linechart(
    data = equilibre_mensuel[, c("date", "consommation", "production")],
    type = "spline"
  ) %>%  
  bb_x_axis(tick = list(format = "%Y-%m", fit = FALSE)) %>%
  bb_zoom(enabled = TRUE)
```
Map variables on the chart

Description
Map variables on the chart

Usage
bb_aes(bb, ...)
bb_aes_string(bb, ...)
bbaes(...)
bbaes_string(...)

Arguments
bb A billboard htmlwidget object.
... Mapping parameters, such as x for x-axis, y for y-axis, group for grouping variable.

Value
A billboard htmlwidget object.

Note
bb_aes is intended to use in a "piping" way. bbaes is the equivalent to use inside a helper function such as bb_barchart, bb_scatterplot...

Examples
## Not run:
dat <- as.data.frame(table(sample(letters[1:5], 100, TRUE)))

billboarder(data = dat) %>%
  bb_aes(x = Var1, y = Freq) %>%
  bb_barchart()

tab <- table(sample(letters[1:5], 100, TRUE), sample(LETTERS[1:5], 100, TRUE))
dat_group <- as.data.frame(tab)

billboarder(data = dat_group) %>%
  bb_aes(x = Var1, y = Freq, group = "Var2") %>%
  bb_barchart()
billboard-theme

Set theme and default colors for Billboard charts

Description

Set theme and default colors for Billboard charts

Usage

```r
set_theme(name = c("billboard", "insight", "graph", "datalab"))
set_color_palette(colors)
```

Arguments

- **name**
  Name of the theme, possible values are: "billboard", "insight", "graph", "datalab".
- **colors**
  Vector of colors to use as default.

Note

You can only use one theme and palette at a time (in Shiny applications or Markdown documents).

Examples

```r
library("billboarder")
set_theme("insight")
data("prod_par_filiere")
billboarder() %>%
  bb_barchart(
    data = prod_par_filiere[, c("annee", "prod_hydraulique", "prod_eolien", "prod_solaire")]
  ) %>%
  bb_data(
    names = list(prod_hydraulique = "Hydraulic", prod_eolien = "Wind", prod_solaire = "Solar")
  ) %>%
  bb_y_grid(show = TRUE) %>%
  bb_y_axis(tick = list(format = suffix("TWh")),
            label = list(text = "production (in terawatt-hours)", position = "outer-top")) %>%
  bb_legend(position = "inset", inset = list(anchor = "top-right")) %>%
  bb_labs(title = "Renewable energy production",
          caption = "Data source: RTE (https://opendata.rte-france.com")
)
Create a Billboard.js widget

Description

Create an interactive visualization with Javascript library Billboard.js

Usage

```r
billboarder(bbillboarder(
    bb_opts = list(),
    data = NULL,
    width = NULL,
    height = NULL,
    elementId = NULL
)
```

Arguments

- `bb_opts` A list in JSON format with chart parameters, see https://naver.github.io/billboard.js/demo/.
- `data` A data.frame.
- `width` A numeric input in pixels.
- `height` A numeric input in pixels.
- `elementId` Use an explicit element ID for the widget.

Description

The following functions are imported and then re-exported from the billboarder package to avoid listing the magrittr as Depends of billboarder
bbleboarder-shiny

Shiny bindings for billboarder

Description

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

Usage

```r
bbboarderOutput(outputId, width = "100\%", height = "400px")
renderBillboarder(expr, env = parent.frame(), quoted = FALSE)
bbboarderProxy(
  shinyId,
  data = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

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 billboarder
- `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.
- `shinyId`: single-element character vector indicating the output ID of the chart to modify (if invoked from a Shiny module, the namespace will be added automatically)
- `data`: A data.frame.
- `session`: the Shiny session object to which the chart belongs; usually the default value will suffice

See Also

`proxy_example`

Examples

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

  ui <- fluidPage(
    tags$h2("Include billboard charts in Shiny"),
```
```r
fluidRow(
  column(
    width = 6,
    billboarderOutput("mybb1"),
    tags$p("Click on a bar to get the value: "),
    verbatimTextOutput("res_click")
  ),
  column(
    width = 6,
    billboarderOutput("mybb2")
  )
)

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

  output$mybb1 <- renderBillboarder(
    dat <- data.frame(
      label = paste("Label", 1:5),
      value = sample.int(100, 5)
    )
    billboarder() %>%
    bb_barchart(
      data = dat,
      mapping = bbaes(label, value),
      rotated = TRUE
    )
  )

  output$res_click <- renderPrint(
    input$mybb1_click
  )

  output$mybb2 <- renderBillboarder(
    data(AirPassengers)
    air_passengers <- data.frame(
      date = as.Date(paste(
        rep(1949:1960, each = 12),
        rep(1:12, times = 12),
        "01", sep = "-
      ),
      passengers = AirPassengers
    )
    billboarder() %>%
    bb_linechart(
      data = air_passengers,
      mapping = bbaes(date, passengers), type = "spline"
    )
  )
}
```
cdc_prod_filiere  

French electricity generation by power source for the day of 2017-06-12.

Description

Average power generation (MW) per 30-minute interval within continental France, aggregated by broad power source. Last update: 2017-07-27.

Usage

cdc_prod_filiere

Format

A data frame with 48 rows and 11 variables:

date_heure  Timestamp (POSIXct)
prod_total  Total production in MW (thermal + hydro + nuclear + solar + wind + bioenergy)
prod_gaz    Gas production in MW
prod_bioenergies  Bioenergy production in MW
prod_hydraulique  Hydraulic production in MW
prod_thermique_fossile  Fossil thermal production in MW
prod_charbon  Coal production in MW
prod_eolien  Wind production in MW
prod_solaire  Solar production in MW
prod_nucleaire  Nuclear production in MW
prod_fioul    Oil production in MW

Source

RTE (https://opendata.reseaux-energies.fr/explore/dataset/production-quotidienne-filiere)
Description

Monthly history of supply/demand balance (GWh) based on gross consumption, the balance of physical exchanges with foreign countries and offtakes due to pumping. Last update: 2017-07-27.

Usage

equilibre_mensuel

Format

A data frame with 126 rows and 5 variables:

date Date
solde Supply/demand balance (in GWh)
production Generation (in GWh)
pompage Pumping for hydraulic generation (in GWh)
consommation Consumption (in GWh)

Source

RTE (https://opendata.reseaux-energies.fr/explore/dataset/equilibre-national-mensuel-prod-conso-brute/)

prefix

Shortcut to add a prefix value to axis labels

Description

Shortcut to add a prefix value to axis labels

Usage

prefix(x)

Arguments

x A character of length one.

See Also

suffix
prod_filiere_long  French electricity generation by year and branch.

Description

Annual French electricity production (TWh) by branch. Last update: 2017-02-15.

Usage

prod_filiere_long

Format

A data frame with 45 rows and 3 variables:

- **annee**: Year
- **branche**: Source of production
- **prod**: Production in TWh

Source


prod_par_filiere  French electricity generation by year and branch.

Description

Annual French electricity production (TWh) by branch. Last update: 2017-02-15.

Usage

prod_par_filiere

Format

A data frame with 5 rows and 11 variables:

- **annee**: Year
- **prod_total**: Total production in TWh (thermal + hydro + nuclear + solar + wind + bioenergy)
- **prod_therm**: Thermal production in TWh (oil + gas + coal)
- **prod_hydraulique**: Hydraulic production in TWh
- **prod_bioenergies**: Bioenergy production in TWh
- **prod_eolien**: Wind production in TWh
**Description**

Launch an example to demonstrate how to use proxy method from `billboarder` in Shiny app.

**Usage**

```r
proxy_example(chart = "gauge")
```

**Arguments**

- `chart`: Chart type for which to see an example, possible values are `gauge`, `pie`, `bar`, `bar2`, `line`, `line2`, `density`, `histogram`, `lollipop`, `stacked_bar`.

**Examples**

```r
if (interactive()) {

  # Titanic passenger
  proxy_example("bar")

  # Electricity production by sources and year
  proxy_example("bar2")

  # Moving lollipop with mpg dataset from ggplot2
  proxy_example("lollipop")

  # Update a stacked bar chart
  proxy_example("stacked_bar")

  # Moving sine and cosine
  proxy_example("line")

  # Changing lines and adding ones
  proxy_example("line2")

```
# Update pie chart
proxy_example("pie")

# Density with ggplot2 diamonds
proxy_example("density")

# Histogram with ggplot2 diamonds
proxy_example("histogram")

}

suffix  

Shortcut to add a suffix value to axis labels

Description

Shortcut to add a suffix value to axis labels

Usage

suffix(x)

Arguments

x  
A character of length one.

See Also

prefix
**Index**

<table>
<thead>
<tr>
<th>* datasets</th>
<th>+ datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>avengers, 4</td>
<td>bb_line, 31</td>
</tr>
<tr>
<td>cdc_prod_filiere, 66</td>
<td>bb_linechart, 31</td>
</tr>
<tr>
<td>equilibre_mensuel, 67</td>
<td>bb_load, 34</td>
</tr>
<tr>
<td>prod_filiere_long, 68</td>
<td>bb_lollipop, 35</td>
</tr>
<tr>
<td>prod_par_filiere, 68</td>
<td>bb_pie, 36</td>
</tr>
<tr>
<td>%&gt;% (billboarder-exports), 63</td>
<td>bb_piechart, 37</td>
</tr>
<tr>
<td>aes (billboarder-exports), 63</td>
<td>bb_point, 38</td>
</tr>
<tr>
<td>avengers, 4</td>
<td>bb_proxy_axis_labels, 39</td>
</tr>
<tr>
<td>avengers_wide (avengers), 4</td>
<td>bb_proxy_data_colors, 39</td>
</tr>
<tr>
<td>bauge, 4</td>
<td>bb_proxy_data_names, 41</td>
</tr>
<tr>
<td>bauge-shiny, 6</td>
<td>bb_proxy_defocus (bb_proxy_focus), 44</td>
</tr>
<tr>
<td>baugeOutput (bauge-shiny), 6</td>
<td>bb_proxy_flow, 42</td>
</tr>
<tr>
<td>bb_add_style, 7, 53</td>
<td>bb_proxy_focus, 44</td>
</tr>
<tr>
<td>bb_aes (billboard-aes), 61</td>
<td>bb_proxy_groups, 45</td>
</tr>
<tr>
<td>bb_aes_string (billboard-aes), 61</td>
<td>bb_proxy_hide, 46, 49</td>
</tr>
<tr>
<td>bb_area, 8</td>
<td>bb_proxy_legend, 46</td>
</tr>
<tr>
<td>bb_axis, 9</td>
<td>bb_proxy_show, 46, 48</td>
</tr>
<tr>
<td>bb_bar, 10</td>
<td>bb_proxy_tooltip, 49</td>
</tr>
<tr>
<td>bb_bar_color_manual, 12</td>
<td>bb_proxy_transform, 50</td>
</tr>
<tr>
<td>bb_barchart, 10</td>
<td>bb_proxy_xs, 50</td>
</tr>
<tr>
<td>bb_bubble, 13</td>
<td>bb_radar, 51, 52</td>
</tr>
<tr>
<td>bb_callbacks, 14</td>
<td>bb_radarchart, 52</td>
</tr>
<tr>
<td>bb_categories, 15</td>
<td>bb_regions, 53</td>
</tr>
<tr>
<td>bb_color, 16</td>
<td>bb_scatterplot, 54</td>
</tr>
<tr>
<td>bb_colors_manual, 17</td>
<td>bb_spline, 56</td>
</tr>
<tr>
<td>bb_data, 18</td>
<td>bb_subchart, 56</td>
</tr>
<tr>
<td>bb_densityplot, 19, 27</td>
<td>bb_svg, 57</td>
</tr>
<tr>
<td>bb_donut, 20</td>
<td>bb_title, 57</td>
</tr>
<tr>
<td>bb_donutchart, 21</td>
<td>bb_tooltip, 58</td>
</tr>
<tr>
<td>bb_export, 22</td>
<td>bb_transition, 59</td>
</tr>
<tr>
<td>bb_gauge, 23</td>
<td>bb_unload, 59</td>
</tr>
<tr>
<td>bb_gaugechart, 24</td>
<td>bb_x_axis (bb_axis), 9</td>
</tr>
<tr>
<td>bb_grid, 25</td>
<td>bb_x_grid (bb_grid), 25</td>
</tr>
<tr>
<td>bb_histogram, 19, 26</td>
<td>bb_y_axis (bb_axis), 9</td>
</tr>
<tr>
<td>bb_interaction, 28</td>
<td>bb_y_grid (bb_grid), 25</td>
</tr>
<tr>
<td>bb_labs, 29, 58</td>
<td>bb_zoom, 60</td>
</tr>
<tr>
<td>bb_legend, 30</td>
<td>bbaes, 11, 19, 21, 26, 32, 35, 37, 52, 55</td>
</tr>
<tr>
<td></td>
<td>bbaes (billboard-aes), 61</td>
</tr>
<tr>
<td></td>
<td>bbaes_string (billboard-aes), 61</td>
</tr>
</tbody>
</table>
billboard-aes, 61
billboard-theme, 62
billboarder, 22, 63
billboarder-exports, 63
billboarder-package, 3
billboarder-shiny, 64
billboarderOutput (billboarder-shiny), 64
billboarderProxy, 11, 18, 21, 22, 25, 32, 37, 53, 55
billboarderProxy (billboarder-shiny), 64
cdc_prod_filiere, 66
density, 19
equilibre_mensuel, 67
geom_histogram, 27
JS (billboarder-exports), 63
prefix, 67
prod_filiere_long, 68
prod_par_filiere, 68
proxy_example, 64, 69
renderBauge (bauge-shiny), 6
renderBillboarder (billboarder-shiny), 64
set_color_palette (billboard-theme), 62
set_theme (billboard-theme), 62
suffix, 70