Package 'ECharts2Shiny'

January 20, 2025

Type Package

Title Embedding Interactive Charts Generated with ECharts Library into Shiny Applications

Imports shiny,jsonlite

Version 0.2.13

Date 2017-12-11

Maintainer Xiaodong Deng <xd_deng@hotmail.com>

Description Embed interactive charts to their Shiny applications. These charts will be generated by ECharts library developed by Baidu (<http://echarts.baidu.com/>). Current version supports line chart, bar chart, pie chart, scatter plot, gauge, word cloud, radar chart, tree map, and heat map.

URL https://github.com/XD-DENG/ECharts2Shiny

BugReports https://github.com/XD-DENG/ECharts2Shiny/issues

License GPL-2

LazyData TRUE

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Xiaodong Deng [aut, cre],
Hao Zhu [ctr],
Yiheng Li [ctr],
Janet Wagner [ctr],
ChinYong Lim [ctr]

Repository CRAN

Date/Publication 2017-12-11 14:50:23 UTC

Contents

`deliverChart` *Deliver the Chart in the UI Component of Shiny Applications*

Description

This function helps deliver the charts plotted by ECharts into Shiny applications.

Usage

```
deliverChart(div_id, running_in_shiny = TRUE)
```

Arguments

`div_id`	The id of the div which you need to specify first with tags$div() function of Shiny.
`running_in_shiny`	If we're actually running this in a Shiny library, or we're simply doing testing. Default valus is "TRUE". If "FALSE", the function will print what it's supposed to evaluate.

Details

This will help us deliver the interactive charts. At the back-end, everything is donw by Javascript.

Note

Users need to state the division for the chart first, with tags$div() function of Shiny packages. Please note that the division id must keep unique (duplicated division id will cause error).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)

  # Prepare sample data for plotting --------------------------
  dat <- data.frame(c(1, 2, 3))
  names(dat) <- c("Type-A")
  row.names(dat) <- c("Time-1", "Time-2", "Time-3")

  # Server function -------------------------------------------
  server <- function(input, output) {
    # Call functions from ECharts2Shiny to render charts
    renderBarChart(div_id = "test", grid_left = '1%', direction = "vertical", data = dat)
  }

  # UI layout -------------------------------------------------
  ui <- fluidPage(
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),

    tags$div(id="test", style="width:50%;height:400px;"),
    deliverChart(div_id = "test")
  )

  # Run the application ---------------------------------------
  shinyApp(ui = ui, server = server)
}
```

`loadEChartsLibrary` *Load the Javascript Library File of ECharts to the Shiny Application*

Description

This function will help load the Javascript library file of ECharts to the current shiny project. This is mandatory before we can plot with ECharts in Shiny applications.

Usage

```
loadEChartsLibrary()
```

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)

  # Prepare sample data for plotting --------------------------
  dat <- data.frame(c(1, 2, 3))
  names(dat) <- c("Type-A")
  row.names(dat) <- c("Time-1", "Time-2", "Time-3")

  # Server function -------------------------------------------
  server <- function(input, output) {
    # Call functions from ECharts2Shiny to render charts
    renderBarChart(div_id = "test", grid_left = '1%', direction = "vertical",
                   data = dat)
  }

  # UI layout -------------------------------------------------
  ui <- fluidPage(
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),

    tags$div(id="test", style="width:50%;height:400px;"),
    deliverChart(div_id = "test")
  )

  # Run the application ---------------------------------------
  shinyApp(ui = ui, server = server)
}
```

`loadEChartsTheme` *Load the Theme File of ECharts to the Shiny Application*

Description

This function will help load the theme file of ECharts into the current Shiny application. This is not mandatory for the basic use of this package. But if users want to try different theme, they need to load the corresponding theme file.

Usage

```
loadEChartsTheme(theme)
```

Arguments

`theme`	The theme file users want to use. The valid values include "infographic", "macarons", "roma", "shine", and "vintage".

Details

Users can simply use the default theme. But if they want to try different theme of the charts, they need to load the corresponding JS file.The theme files are not loaded automatically so that we don't have to include unnecessary files into the Shiny applications (you only include what you need).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)

  # Prepare sample data for plotting --------------------------
  dat <- data.frame(c(1, 2, 3))
  names(dat) <- c("Type-A")
  row.names(dat) <- c("Time-1", "Time-2", "Time-3")

  # Server function -------------------------------------------
  server <- function(input, output) {
    # Call functions from ECharts2Shiny to render charts
    renderBarChart(div_id = "test", grid_left = '1%', direction = "vertical",
                   data = dat, theme = "vintage")
  }

  # UI layout -------------------------------------------------
  ui <- fluidPage(
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),
    loadEChartsTheme("vintage"),

    tags$div(id="test", style="width:50%;height:400px;"),
    deliverChart(div_id = "test")
  )

  # Run the application ---------------------------------------
  shinyApp(ui = ui, server = server)
}
```

`renderBarChart` *Render the Bar Chart Plotted by ECharts into Shiny Application*

Description

renderBarChart() function helps render the bar chart into Shiny application.

Usage

```
renderBarChart(div_id, data, theme,
               stack_plot = FALSE, direction = "horizontal",
               grid_left,grid_right, grid_top, grid_bottom,
               show.legend = TRUE, show.tools = TRUE,
               font.size.legend = 12,
               font.size.axis.x = 12, font.size.axis.y = 12,
               axis.x.name = NULL, axis.y.name = NULL,
               rotate.axis.x = 0, rotate.axis.y = 0,
               bar.max.width = NULL,
               animation = TRUE,
               hyperlinks = NULL,
               running_in_shiny)
```

Arguments

div_id	The division id users specified for this chart. The division will be specified in ui.R.
data	The data used for the plotting. It should be a data.frame. Each column of the data.frame is one category, and each row is one observation (like one timepoint).
theme	Which ECharts theme to use. Valid values include "default", "roma", "infographic", "macarons", "vintage", "shine", "caravan", "dark-digerati", "jazz", and "london".
stack_plot	Whether do stack bar chart. The default value is FALSE.
direction	The direction of the bar chart. Valid values include "vertical" and "horizontal". Default value is "horizontal".
grid_left	Distance between grid component and the left side of the container. Default value is "3%".
grid_right	Distance between grid component and the right side of the container. Default value is "4%".
grid_top	Distance between grid component and the top side of the container. Default value is "16%".
grid_bottom	Distance between grid component and the bottom side of the container. Default value is "3%".
show.legend	If display the legends. The default value is TRUE.
show.tools	If display the tool bar. The default value is TRUE.
font.size.legend	The font size of legend bar. The default value is 12.
font.size.axis.x	The font size of the labels on X axis. The default value is 12.
font.size.axis.y	The font size of the labels on Y axis. The default value is 12.
axis.x.name	The name of X axis. The default value is NULL.
axis.y.name	The name of Y axis. The default value is NULL.

`rotate.axis.x` The rotation degree of labels on X axis. The default value is 0.

`rotate.axis.y` The rotation degree of labels on Y axis. The default value is 0.

`bar.max.width` The maximum width of the bar. The default value is NULL, in which case the bar width and maximum width will be automatically adjusted.

Users can also assign a numeric value to it, to customize the maximum bar width. If the width is too big or invalid value (like a character string), it will be automatically adjusted too.

`animation` Whether display the chart with animation. The default value is TRUE.

`hyperlinks` Vector. Users can link each element in the chart to a hyperlink (URL like http://***.com). Please note the length of the "hyperlinks" vector should be the same to the number of rows in the data given.

Note that if hyperlinks are available, the fonts in the pop-up window will be in skyblue color and italic style.

`running_in_shiny`

If we're actually running this in a Shiny library, or we're simply doing testing. Default valus is "TRUE". If "FALSE", the function will print what it's supposed to evaluate.

Note

Users need to state the division for the chart first, with tags$div() function of Shiny packages. Please note that the division id must keep unique (duplicated division id will cause error).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)

  # Prepare sample data for plotting ---------------------------
  dat <- data.frame(c(1, 2, 3),
                    c(2, 4, 6))
  names(dat) <- c("Type-A", "Type-B")
  row.names(dat) <- c("Time-1", "Time-2", "Time-3")

  # Server function --------------------------------------------
  server <- function(input, output) {
    # Call functions from ECharts2Shiny to render charts
    renderBarChart(div_id = "test", grid_left = '1%', direction = "vertical",
                   data = dat)
  }

  # UI layout --------------------------------------------------
  ui <- fluidPage(
```

```
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),

    tags$div(id="test", style="width:50%;height:400px;"),
    deliverChart(div_id = "test")
  )

  # Run the application ------------------------------------
  shinyApp(ui = ui, server = server)
}
```

---

renderGauge *Render the Gauge Chart Plotted by ECharts into Shiny Application*

Description

renderGauge() function helps render the gauge chart into Shiny application.

Usage

```
renderGauge(div_id, theme = "default", gauge_name, rate,
            show.tools = TRUE,
            animation = TRUE,
            running_in_shiny = TRUE)
```

Arguments

div_id	The division id users specified for this chart. The division will be specified in ui.R.
theme	Which ECharts theme to use. Valid values include "default", "roma", "infographic", "macarons", "vintage", "shine", "caravan", "dark-digerati", "jazz", and "london".
gauge_name	The title to show on the gauge. It can not be ignored.
rate	As the gauge helps show some kind of rate, users need to give this rate value. It must be numerical or integer values.
show.tools	If display the tool bar. The default value is TRUE.
animation	Whether display the chart with animation. The default value is TRUE.
running_in_shiny	If we're actually running this in a Shiny library, or we're simply doing testing. Default valus is "TRUE". If "FALSE", the function will print what it's supposed to evaluate.

Note

Users need to state the division for the chart first, with tags$div() function of Shiny packages. Please note that the division id must keep unique (duplicated division id will cause error).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)


  # Server function ------------------------------------------
  server <- function(input, output) {
    # Call functions from ECharts2Shiny to render charts
    renderGauge(div_id = "test",rate = 99, gauge_name = "Finish Rate")
  }

  # UI layout ------------------------------------------------
  ui <- fluidPage(
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),

    tags$div(id="test", style="width:50%;height:400px;"),
    deliverChart(div_id = "test")
  )

  # Run the application --------------------------------------
  shinyApp(ui = ui, server = server)
}
```

`renderHeatMap` *Render Heat Map Plotted by ECharts into Shiny Applications*

Description

renderHeatMap() function helps render heat map charts into Shiny applications.

Usage

```
renderHeatMap(div_id, data,
              theme = "default",
              show.tools = TRUE,
         grid_left = "3%", grid_right = "4%", grid_top = "16%", grid_bottom = "3%",
              running_in_shiny = TRUE)
```

Arguments

div_id	The division id users specified for this chart. The division will be specified in ui.R.
data	The data input must be a matrix containing numeric or integer values. Users can choose to have or not have row names or columns names. From version 2.10, the data input (matrix) must be with row names and column names.
theme	Which ECharts theme to use. Valid values include "default", "roma", "infographic", "macarons", "vintage", "shine", "caravan", "dark-digerati", "jazz", and "london".
show.tools	If display the tool bar. The default value is TRUE.
grid_left	Distance between grid component and the left side of the container. Default value is "3%".
grid_right	Distance between grid component and the right side of the container. Default value is "4%".
grid_top	Distance between grid component and the top side of the container. Default value is "16%".
grid_bottom	Distance between grid component and the bottom side of the container. Default value is "3%".
running_in_shiny	If we're actually running this in a Shiny library, or we're simply doing testing. Default valus is "TRUE". If "FALSE", the function will print what it's supposed to evaluate.

Note

Users need to state the division for the chart first, with tags$div() function of Shiny packages. Please note that the division id must keep unique (duplicated division id will cause error).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

References

https://github.com/ecomfe/echarts-wordcloud

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)


  # Server function ------------------------------------------
  server <- function(input, output) {
```

```
    dat <- volcano
    row.names(dat) <- 1:dim(dat)[1]
    colnames(dat) <- 1:dim(dat)[2]

    renderHeatMap(div_id = "test",
                  data = dat)
  }

  # UI layout ------------------------------------------------
  ui <- fluidPage(
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),

    tags$div(id="test", style="width:50%;height:400px;"),
    deliverChart(div_id = "test")
  )

  # Run the application --------------------------------------
  shinyApp(ui = ui, server = server)
}
```

---

renderLineChart *Render the Line Chart Plotted by ECharts into Shiny Application*

Description

renderLineChart() function helps render the line chart into Shiny application.

Usage

```
renderLineChart(div_id, data, theme = "default",
                line.width = 2, line.type = "solid",
                point.size = 5, point.type = "emptyCircle",
                stack_plot = FALSE, step = "null",
                show.legend = TRUE, show.tools = TRUE,
                font.size.legend= 12,
                font.size.axis.x = 12, font.size.axis.y = 12,
                axis.x.name = NULL, axis.y.name = NULL,
                rotate.axis.x = 0, rotate.axis.y = 0,
                show.slider.axis.x = FALSE, show.slider.axis.y = FALSE,
                animation = TRUE,
           grid_left = "3%", grid_right = "4%", grid_top = "16%", grid_bottom = "3%",
                running_in_shiny = TRUE)
```

Arguments

div_id	The division id users specified for this chart. The division will be specified in ui.R.

`data` The data used for the plotting. It should be a data.frame. Each column of the data.frame is one category, and each row is one observation (like one timepoint).

`theme` Which ECharts theme to use. Valid values include "default", "roma", "infographic", "macarons", "vintage", "shine", "caravan", "dark-digerati", "jazz", and "london".

`line.width` This is to help set the width of the lines.

The value should be either numeric or integer. The default value is 2.

Its length should be either one or the same as the number of categories in the data (the number of columns in the data).

`line.type` The type of the lines.

The value can be "solid", "dashed", or "dotted". The default value is "solid".

Its length should be either one or the same as the number of categories in the data (the number of columns in the data).

`point.size` This argument helps set the size of points in the scatter plots.

The value should be either numeric or integer. The default value is 5.

Its length should be either one or the same as the number of categories in the data (the number of columns in the data).

`point.type` The shape of the points in scatter plots.

Valid values include 'emptyCircle', 'circle', 'rect', 'roundRect', 'triangle', 'diamond', 'pin', 'arrow'. The default value is 'emptyCircle'.

Its length should be either one or the same as the number of categories in the data (the number of columns in the data).

`stack_plot` Whether do stack line chart. The default value is FALSE.

`step` This argument helps plot step line charts. The default value is "null", i.e., non-step line chart.

If users want step line chart, they can choose "start", "middle", or "end" to determine the turning point positions in the step line charts.

`show.legend` If display the legends. The default value is TRUE.

`show.tools` If display the tool bar. The default value is TRUE.

`font.size.legend`
The font size of legend bar. The default value is 12.

`font.size.axis.x`
The font size of the labels on X axis. The default value is 12.

`font.size.axis.y`
The font size of the labels on Y axis. The default value is 12.

`axis.x.name` The name of X axis. The default value is NULL.

`axis.y.name` The name of Y axis. The default value is NULL.

`rotate.axis.x` The rotation degree of labels on X axis. The default value is 0.

`rotate.axis.y` The rotation degree of labels on Y axis. The default value is 0.

`show.slider.axis.x`
Whether display slider on X axis. The default value is FALSE.

`show.slider.axis.y`
Whether display slider on Y axis. The default value is FALSE.

animation	Whether display the chart with animation. The default value is TRUE.
grid_left	Distance between grid component and the left side of the container. Default value is "3%".
grid_right	Distance between grid component and the right side of the container. Default value is "4%".
grid_top	Distance between grid component and the top side of the container. Default value is "16%".
grid_bottom	Distance between grid component and the bottom side of the container. Default value is "3%".
running_in_shiny	If we're actually running this in a Shiny library, or we're simply doing testing. Default valus is "TRUE". If "FALSE", the function will print what it's supposed to evaluate.

Note

Users need to state the division for the chart first, with tags$div() function of Shiny packages. Please note that the division id must keep unique (duplicated division id will cause error).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)


  dat <- data.frame(c(1, 2, 3, 1),
                    c(2, 4, 6, 6),
                    c(3, 2, 7, 5))
  names(dat) <- c("Type-A", "Type-B", "Type-C")
  row.names(dat) <- c("Time-1", "Time-2", "Time-3", "Time-4")

  # Server function ---------------------------------------
  server <- function(input, output) {
    renderLineChart(div_id = "test",
                    data = dat)
  }

  # UI layout ---------------------------------------------
  ui <- fluidPage(
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),

    tags$div(id="test", style="width:50%;height:400px;"),
    deliverChart(div_id = "test")
```

```
  )

  # Run the application ------------------------------------
  shinyApp(ui = ui, server = server)
}
```

`renderPieChart` *Render the Pie Chart Plotted by ECharts into Shiny Application*

Description

renderPieChart() function helps render the pie chart into Shiny application.

Usage

```
renderPieChart(div_id, data,
               theme = 'default', radius, center_x,center_y,
               show.label = TRUE,
               show.legend = TRUE, show.tools = TRUE,
               font.size.legend= 12,
               animation = TRUE,
               hyperlinks = NULL,
               running_in_shiny)
```

Arguments

`div_id`	The division id users specified for this chart. The division will be specified in ui.R.
`data`	The data used for the plotting. It should be either a vector or a data.frame. If it's a vector, it should be made up of all the elements you want to count and plot, like c("a", "a", "b", "a", "b", "c"). If it's a data.frame, the data must be made up of only two columns, "name" and "value". The "value" column must be numeric or integer.
`theme`	Which ECharts theme to use. Valid values include "default", "roma", "infographic", "macarons", "vintage", "shine", "caravan", "dark-digerati", "jazz", and "london".
`radius`	The radius of the pie chart. The default value is "75%".
`center_x`	The position of the center of the pie chart (x axis). Default value is "50%".
`center_y`	The position of the center of the pie chart (y axis). Default value is "50%".
`show.label`	Whether display the leble for the pie chart. The default value is TRUE.
`show.legend`	Whether display the legends. The default value is TRUE.
`show.tools`	Whether display the tool bar. The default value is TRUE.
`font.size.legend`	The font size of legend bar. The default value is 12.

`animation` Whether display the chart with animation. The default value is TRUE.

`hyperlinks` Vector. Users can link each element in the chart to a hyperlink (URL like http://***.com). Please note this is only supported when the data is in data.frame format, and the length of the "hyperlinks" vector should be the same to the number of rows in the data given.

Note that if hyperlinks are available, the fonts in the pop-up window will be in skyblue color and italic style.

`running_in_shiny`

If we're actually running this in a Shiny library, or we're simply doing testing. Default valus is "TRUE". If "FALSE", the function will print what it's supposed to evaluate.

Note

Users need to state the division for the chart first, with tags$div() function of Shiny packages. Please note that the division id must keep unique (duplicated division id will cause error).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)


  dat <- c(rep("Type-A", 8),
             rep("Type-B", 5),
             rep("Type-C", 1))


  # Server function ---------------------------------------------
  server <- function(input, output) {
    renderPieChart(div_id = "test",
                   data = dat)
  }

  # UI layout ---------------------------------------------------
  ui <- fluidPage(
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),

    tags$div(id="test", style="width:50%;height:400px;"),
    deliverChart(div_id = "test")
  )

  # Run the application -----------------------------------
```

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

`renderRadarChart` *Render the Radar Chart Plotted by ECharts into Shiny Application*

Description

renderRadarChart() function helps render the Radar chart into Shiny application.

Usage

```
renderRadarChart(div_id, data, theme = "default",
                  shape = "default", line.width = 2,
                  show.legend = TRUE, show.tools = TRUE,
                  font.size.legend = 12,
                  animation = TRUE,
                  running_in_shiny = TRUE)
```

Arguments

`div_id`	The division id users specified for this chart. The division will be specified in ui.R.
`data`	The data used for the plotting. It should be a data.frame. For radar chart, the data must have row names and column names specified.
`theme`	Which ECharts theme to use. Valid values include "default", "roma", "infographic", "macarons", "vintage", "shine", "caravan", "dark-digerati", "jazz", and "london".
`shape`	The shape of the radar chart. Valid values include "default" and "circle".
`line.width`	The width of the lines in the radar chart.
`show.legend`	If display the legends. The default value is TRUE.
`show.tools`	If display the tool bar. The default value is TRUE.
`font.size.legend`	The font size of legend bar. The default value is 12.
`animation`	Whether display the chart with animation. The default value is TRUE.
`running_in_shiny`	If we're actually running this in a Shiny library, or we're simply doing testing. Default valus is "TRUE". If "FALSE", the function will print what it's supposed to evaluate.

Note

Users need to state the division for the chart first, with tags$div() function of Shiny packages. Please note that the division id must keep unique (duplicated division id will cause error).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)


  dat <- data.frame(Type.A = c(4300, 10000, 25000, 35000, 50000),
                    Type.B = c(5000, 14000, 28000, 31000, 42000),
                    Type.C = c(4000, 2000, 9000, 29000, 35000))
  row.names(dat) <- c("Feture 1", "Feature 2", "Feature 3", "Feature 4", "Feature 5")


  # Server function ------------------------------------------
  server <- function(input, output) {
    renderRadarChart(div_id = "test",
                    data = dat)
  }

  # UI layout ------------------------------------------------
  ui <- fluidPage(
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),

    tags$div(id="test", style="width:50%;height:400px;"),
    deliverChart(div_id = "test")
  )

  # Run the application --------------------------------------
  shinyApp(ui = ui, server = server)
}
```

---

renderScatter *Render the Scatter Plots Plotted by ECharts into Shiny Application*

Description

renderScatter() function helps render the scatter plots into Shiny applications.

Usage

```
renderScatter(div_id, data,
              point.size = 10, point.type = "circle",
              theme = "default", auto.scale = TRUE,
              show.legend = TRUE, show.tools = TRUE,
```

```
        font.size.legend = 12,
        font.size.axis.x = 12, font.size.axis.y = 12,
        axis.x.name = NULL, axis.y.name = NULL,
        rotate.axis.x = 0, rotate.axis.y = 0,
        show.slider.axis.x = FALSE, show.slider.axis.y = FALSE,
        animation = TRUE,
    grid_left = "3%", grid_right = "4%", grid_top = "16%", grid_bottom = "3%",
        running_in_shiny = TRUE)
```

Arguments

div_id	The division id users specified for this chart. The division will be specified in ui.R.
data	The data used for the plotting. It should be a data.frame. For scatter plots, the data must be made up of three columns, "x", "y", and "group".
point.size	This argument helps set the size of points in the scatter plots. It should be a single numeric or integer value. The dafault value is 10.
point.type	The shape of the points in scatter plots. Valid values include 'emptyCircle', 'circle', 'rect', 'roundRect', 'triangle', 'diamond', 'pin', 'arrow'. The default value is 'circle'. The length of this argument should either be one or be the same as the number of UNIQUE groups (the number of unique elements in 'group' column of the data input).
theme	Which ECharts theme to use. Valid values include "default", "roma", "infographic", "macarons", "vintage", "shine", "caravan", "dark-digerati", "jazz", and "london".
auto.scale	A logical argument to determine if the scatter plot should be scaled again automatically after the users exclude any group of observations. The default value is TRUE.
show.legend	If display the legends. The default value is TRUE.
show.tools	If display the tool bar. The default value is TRUE.
font.size.legend	The font size of legend bar. The default value is 12.
font.size.axis.x	The font size of the labels on X axis. The default value is 12.
font.size.axis.y	The font size of the labels on Y axis. The default value is 12.
axis.x.name	The name of X axis. The default value is NULL.
axis.y.name	The name of Y axis. The default value is NULL.
rotate.axis.x	The rotation degree of labels on X axis. The default value is 0.
rotate.axis.y	The rotation degree of labels on Y axis. The default value is 0.
show.slider.axis.x	Whether display slider on X axis. The default value is FALSE.
show.slider.axis.y	Whether display slider on Y axis. The default value is FALSE.

animation	Whether display the chart with animation. The default value is TRUE.
grid_left	Distance between grid component and the left side of the container. Default value is "3%".
grid_right	Distance between grid component and the right side of the container. Default value is "4%".
grid_top	Distance between grid component and the top side of the container. Default value is "16%".
grid_bottom	Distance between grid component and the bottom side of the container. Default value is "3%".
running_in_shiny	If we're actually running this in a Shiny library, or we're simply doing testing. Default valus is "TRUE". If "FALSE", the function will print what it's supposed to evaluate.

Note

Users need to state the division for the chart first, with tags$div() function of Shiny packages. Please note that the division id must keep unique (duplicated division id will cause error).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)


  dat <- data.frame(x = iris$Sepal.Length,
                    y = iris$Sepal.Width,
                    group = iris$Species)


  # Server function ---------------------------------------------
  server <- function(input, output) {
    renderScatter("test", data = dat)
  }

  # UI layout ---------------------------------------------------
  ui <- fluidPage(
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),

    tags$div(id="test", style="width:100%;height:500px;"),
    deliverChart(div_id = "test")
  )
```

```
  # Run the application ------------------------------------
  shinyApp(ui = ui, server = server)
}
```

`renderTreeMap` *Render Interactive Tree Map into Shiny Application*

Description

renderTreeMap() function helps render interactive tree map chart into Shiny application.

Usage

```
renderTreeMap(div_id,
              data,
              name = "Main",
              leafDepth = 2,
              theme = "default",
              show.tools = TRUE,
              running_in_shiny = TRUE)
```

Arguments

`div_id`	The division id users specified for this chart. The division will be specified in ui.R.
`data`	The data used for tree map. Not like other charts, here we need to use JSON data for the tree map charts, since there may be nested data which can be handled by JSON much easilier (please check 'example').
`name`	The name of the tree map chart.
`leafDepth`	It determines when the 'drill down' feature will start to work. This is for mainly nested data, like we may ave categories 'A-1' and 'A-2' under 'A', and furtherly we have 'A-1-1' and 'A-1-2' under 'A-1'. The value of this argument must be integer and bigger than 0 (float value will be converted to its ceiling value automatically). The default value is 2.
`theme`	Which ECharts theme to use. Valid values include "default", "roma", "infographic", "macarons", "vintage", "shine", "caravan", "dark-digerati", "jazz", and "london".
`show.tools`	If display the tool bar. The default value is TRUE.
`running_in_shiny`	If we're actually running this in a Shiny library, or we're simply doing testing. Default valus is "TRUE". If "FALSE", the function will print what it's supposed to evaluate.

Note

Users need to state the division for the chart first, with tags$div() function of Shiny packages. Please note that the division id must keep unique (duplicated division id will cause error).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

Examples

```
if (interactive()) {
    library(shiny)
    library(ECharts2Shiny)

    # Prepare sample data for plotting --------------------------
    dat <- "[{name: 'A',
              value: 6,
              children: [
                  {
                  name: 'A-1',
                  value: 6,
                  children:[
                  {
                  name: 'A-1-1',
                  value: 6
                  },
                  {
                  name: 'A-1-2',
                  value: 2
                  }
                  ]
                  },
                  {
                  name: 'A-2',
                  value: 3
                  }
              ]
            },
            {
              name: 'B',
              value: 6,
              children: [
                  {name : 'B-1',
                  value:10
                  },
                  {
                  name:'B-2',
                  value:2
                  }
              ]
```

```
            },
            {
              name: 'C',
              value: 4
            }]"

    # Server function ---------------------------------------------
    server <- function(input, output) {
      # Call functions from ECharts2Shiny to render charts
      renderTreeMap(div_id = "test",
                    data = dat)
        }

    # UI layout ---------------------------------------------------
    ui <- fluidPage(
      # We MUST load the ECharts javascript library in advance
      loadEChartsLibrary(),

      tags$div(id="test", style="width:100%;height:500px;"),
      deliverChart(div_id = "test")
    )

    # Run the application -----------------------------------
    shinyApp(ui = ui, server = server)

}
```

---

renderWordcloud *Render the Word Cloud Plotted by ECharts into Shiny Application*

Description

renderWordcloud() function helps render the word cloud charts into Shiny applications.

Usage

```
renderWordcloud(div_id, data,
                shape = 'circle',
                grid_size = 5, sizeRange = c(15, 50),
                rotationRange = c(-45, 45),
                hyperlinks = NULL,
                running_in_shiny = TRUE)
```

Arguments

div_id	The division id users specified for this chart. The division will be specified in ui.R.

`data` The data used for the plotting. It should be either a vector or a data.frame.

If it's a vector, it should be made up of all the elements you want to count and plot, like c("a", "a", "b", "a", "b", "c").

If it's a data.frame, the data must be made up of only two columns, "name" and "value". The "value" column must be numeric or integer.

`shape` The shape of the word cloud. The valid values include "circle" (default value), "cardioid", "diamond", "triangle-forward", "triangle", "pentagon" and "star".

`grid_size` The size of the grid in word cloud.

`sizeRange` The font size range in the word cloud. It should be a vector of length two. The default value is c(15, 50).

`rotationRange` The rotation angle range in the word cloud. It should be a vector of length two. The default value is c(-45, 45).

`hyperlinks` Vector. Users can link each element in the chart to a hyperlink (URL like http://***.com). Please note this is only supported when the data is in data.frame format, and the length of the "hyperlinks" vector should be the same to the number of rows in the data given.

Note that if hyperlinks are available, the fonts in the pop-up window will be in skyblue color and italic style.

`running_in_shiny`

If we're actually running this in a Shiny library, or we're simply doing testing. Default valus is "TRUE". If "FALSE", the function will print what it's supposed to evaluate.

Note

Users need to state the division for the chart first, with tags$div() function of Shiny packages. Please note that the division id must keep unique (duplicated division id will cause error).

Author(s)

Xiaodong DENG

(ECharts library is authored by Baidu team)

References

https://github.com/ecomfe/echarts-wordcloud

Examples

```
if (interactive()) {
  library(shiny)
  library(ECharts2Shiny)


  sample_data_for_wordcloud <- c(rep("R", 100),
                                 rep("Python", 100),
                                 rep("SAS", 90),
                                 rep("VBA", 50))
```

```
  # Server function ------------------------------------------
  server <- function(input, output) {
    renderWordcloud("test", data =sample_data_for_wordcloud,
                    grid_size = 10, sizeRange = c(20, 50))
  }

  # UI layout ------------------------------------------------
  ui <- fluidPage(
    # We MUST load the ECharts javascript library in advance
    loadEChartsLibrary(),

    tags$div(id="test", style="width:100%;height:500px;"),
    deliverChart(div_id = "test")
  )

  # Run the application --------------------------------------
  shinyApp(ui = ui, server = server)
}
```

Index