Package 'RLumShiny'

November 14, 2016

Type Package

Title 'Shiny' Applications for the R Package 'Luminescence'

Version 0.1.1

Date 2016-07-20

Author Christoph Burow [aut, cre],
R Luminescence Package Team [ctb],
Jan Odvarko [cph] (jscolor.js in www/jscolor),
AnalytixWare [cph] (ShinySky package)

Maintainer Christoph Burow <christoph.burow@uni-koeln.de>

Description A collection of 'shiny' applications for the R package 'Luminescence'. These mainly, but not exclusively, include applications for plotting chronometric data from e.g. luminescence or radiocarbon dating. It further provides access to bootstraps tooltip and popover functionality and contains the 'jscolor.js' library with a custom 'shiny' output binding.

License GPL-3

Depends R (>= 3.3.0)

Imports Luminescence (>= 0.6.1), shiny (>= 0.13.0), googleVis

URL https://github.com/tzerk/RLumShiny

BugReports https://github.com/tzerk/RLumShiny/issues

Collate 'app_RLum.R' 'jscolor.R' 'tooltip.R' 'popover.R' 'RLumShiny.R' 'zzz.R'

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-07-20 17:49:32

R topics documented:

RLumShiny-package *Shiny Applications for the R Package Luminescence*

Description

A collection of shiny applications for the R package Luminescence. These mainly, but not exclusively, include applications for plotting chronometric data from e.g. luminescence or radiocarbon dating. It further provides access to bootstraps tooltip and popover functionality as well as a binding to JSColor.

Details

In addition to its main purpose of providing convenient access to the Luminescence shiny applications (see app_RLum) this package also provides further functions to extend the functionality of shiny. From the Bootstrap framework the JavaScript tooltip and popover components can be added to any shiny application via tooltip and popover. It further provides a custom input binding to the JavaScript/HTML color picker JSColor. Offering access to most options provided by the JSColor API the function jscolorInput is easily implemented in a shiny app. RGB colors are returned as hex values and can be directly used in R's base plotting functions without the need of any format conversion.

---

app_RLum *Run Luminescence shiny apps*

Description

A wrapper for runApp to start interactive shiny apps for the R package Luminescence.

Usage

```
app_RLum(app, ...)
```

Arguments

app	character (required): name of the application to start. See details for a list of available apps.
...	further arguments to pass to runApp

Details

The RLumShiny package provides a single function from which all shiny apps can be started: app_RLum(). It essentially only takes one argument, which is a unique keyword specifying which application to start. See the table below for a list of available shiny apps and which keywords to use.

Application name:	**Keyword:**	**Function:**
Abanico Plot	*abanico*	plot_AbanicoPlot
Histogram	*histogram*	plot_Histogram
Kernel Density Estimate Plot	*KDE*	plot_KDE
Radial Plot	*radialplot*	plot_RadialPlot

Dose Recovery Test	*doserecovery*	plot_DRTResults
Cosmic Dose Rate	*cosmicdose*	calc_CosmicDoseRate
CW Curve Transformation	*transformCW*	CW2pHMi, CW2pLM, CW2pLMi, CW2pPMi

The app_RLum() function is just a wrapper for runApp. Via the ... argument further arguments can be directly passed to runApp. See ?shiny::runApp for further details on valid arguments.

Author(s)

Christoph Burow, University of Cologne (Germany)

See Also

runApp

Examples

```
## Not run:
# Plotting apps
app_RLum("abanico")
app_RLum("histogram")
app_RLum("KDE")
app_RLum("radialplot")
app_RLum("doserecovery")

# Further apps
app_RLum("cosmicdose")

## End(Not run)
```

---

jscolorInput *Create a JSColor picker input widget*

Description

Creates a JSColor (Javascript/HTML Color Picker) widget to be used in shiny applications.

Usage

```
jscolorInput(inputId, label, value, position = "bottom",
  color = "transparent", mode = "HSV", slider = TRUE, close = FALSE)
```

Arguments

inputId	character (required): Specifies the input slot that will be used to access the value.
label	character: Display label for the control, or NULL for no label.
value	character: Initial RGB value of the color picker. Default is black ('#000000').
position	character: Position of the picker relative to the text input ('bottom', 'left', 'top', 'right').

color	character: Picker color scheme ('transparent' by default). Use RGB color coding ('000000').
mode	character: Mode of hue, saturation and value. Can either be 'HSV' or 'HVS'.
slider	logical: Show or hide the slider.
close	logical: Show or hide a close button.

See Also

Other input.elements: animationOptions, sliderInput; checkboxGroupInput; checkboxInput; dateInput; dateRangeInput; fileInput; numericInput; passwordInput; radioButtons; selectInput, selectizeInput; submitButton; textInput

Examples

```
# html code
jscolorInput("col", "Color", "21BF6B", slider = FALSE)

# example app
## Not run:
shinyApp(
ui = fluidPage(
  jscolorInput(inputId = "col", label = "JSColor Picker",
               value = "21BF6B", position = "right",
               mode = "HVS", close = TRUE),
  plotOutput("plot")
),
server = function(input, output) {
  output$plot <- renderPlot({
    plot(cars, col = input$col, cex = 2, pch = 16)
 })
})

## End(Not run)
```

---

popover *Create a bootstrap button with popover*

Description

Add small overlays of content for housing secondary information.

Usage

```
popover(title, content, header = NULL, html = TRUE,
  class = "btn btn-default", placement = c("right", "top", "left",
  "bottom"), trigger = c("click", "hover", "focus", "manual"))
```

Arguments

title	character (required): Title of the button.
content	character: Text to be displayed in the popover.
header	character: Optional header in the popover.
html	logical Insert HTML into the popover.
class	logical Bootstrap button class (e.g. "btn btn-danger").
placement	character: How to position the popover - top | bottom | left | right | auto. When "auto" is specified, it will dynamically reorient the popover. For example, if placement is "auto left", the popover will display to the left when possible, otherwise it will display right.
trigger	character: How popover is triggered - click | hover | focus | manual.

Examples

```
# html code
popover("title", "Some content")

# example app
## Not run:
shinyApp(
ui = fluidPage(
  jscolorInput(inputId = "col", label = "JSColor Picker",
               value = "21BF6B", position = "right",
               mode = "HVS", close = TRUE),
  popover(title = "Help!", content = "Call 911"),
  plotOutput("plot")
),
server = function(input, output) {
  output$plot <- renderPlot({
    plot(cars, col = input$col, cex = 2, pch = 16)
 })
})

## End(Not run)
```

---

tooltip *Create a bootstrap tooltip*

Description

Create bootstrap tooltips for any HTML element to be used in shiny applications.

Usage

```
tooltip(refId, text, attr = NULL, animation = TRUE, delay = 100,
  html = TRUE, placement = "auto", trigger = "hover")
```

Arguments

refId	character (required): id of the element the tooltip is to be attached to.
text	character: Text to be displayed in the tooltip.
attr	character: Attach tooltip to all elements with attribute attr='refId'.
animation	logical: Apply a CSS fade transition to the tooltip.
delay	numeric: Delay showing and hiding the tooltip (ms).
html	logical: Insert HTML into the tooltip.
placement	character: How to position the tooltip - top | bottom | left | right | auto. When 'auto' is specified, it will dynamically reorient the tooltip. For example, if placement is 'auto left', the tooltip will display to the left when possible, otherwise it will display right.
trigger	character: How tooltip is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space.

Examples

```
# javascript code
tt <- tooltip("elementId", "This is a tooltip.")
str(tt)

# example app
## Not run:
shinyApp(
ui = fluidPage(
  jscolorInput(inputId = "col", label = "JSColor Picker",
               value = "21BF6B", position = "right",
               mode = "HVS", close = TRUE),
  tooltip("col", "This is a JScolor widget"),

  checkboxInput("cbox", "Checkbox", FALSE),
  tooltip("cbox", "This is a checkbox"),

  checkboxGroupInput("cboxg", "Checkbox group", selected = "a",
                     choices = c("a" = "a",
                                 "b" = "b",
                                 "c" = "c")),
  tooltip("cboxg", "This is a <b>checkbox group</b>", html = TRUE),

  selectInput("select", "Selectinput", selected = "a", choices = c("a"="a", "b"="b")),
  tooltip("select", "This is a text input field", attr = "for", placement = "right"),

  passwordInput("pwIn", "Passwordinput"),
  tooltip("pwIn", "This is a password input field"),

  plotOutput("plot")
),
server = function(input, output) {
  output$plot <- renderPlot({
    plot(cars, col = input$col, cex = 2, pch = 16)
 })
})

## End(Not run)
```

Index