Package 'RLumShiny'

July 9, 2025

Type Package

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

Version 0.2.5

Date 2025-07-09

Maintainer Christoph Burow <christoph.burow@gmx.net>

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

Encoding UTF-8

Depends R (>= 4.3)

Imports Luminescence (>= 1.1.0),
shiny (>= 1.11.1),
rhandsontable (>= 0.3.8),
data.table (>= 1.17.6),
googleVis (>= 0.7.3),
leaflet (>= 2.2.2),
shinydashboard (>= 0.7.3),
RCarb (>= 0.1.6),
markdown (>= 1.13),
readxl (>= 1.4.5),
DT (>= 0.33),
knitr (>= 1.50)

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

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

RoxygenNote 7.3.2

Contents

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.

Author(s)

Maintainer: Christoph Burow <christoph.burow@gmx.net> (ORCID)

Authors:

- Urs Tilmann Wolpert
- Sebastian Kreutzer (ORCID)
- Marco Colombo (ORCID)

Other contributors:

- R Luminescence Package Team [contributor]
- Jan Odvarko (jscolor.js in www/jscolor) [copyright holder]
- AnalytixWare (ShinySky package) [copyright holder]
- RStudio (chooser_inputBinding.js in www/ and chooser.R in R/) [copyright holder]

See Also

Useful links:

- https://tzerk.github.io/RLumShiny/
- Report bugs at https://github.com/tzerk/RLumShiny/issues

app_RLum *Run Luminescence shiny apps*

Description

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

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. If no keyword is used a dashboard will be started instead, from which an application can be started.

Application name:	**Keyword:**	**Function:**
Abanico Plot	*abanico*	Luminescence::plot_AbanicoPlot
Histogram	*histogram*	Luminescence::plot_Histogram
Kernel Density Estimate Plot	*KDE*	Luminescence::plot_KDE
Radial Plot	*radialplot*	Luminescence::plot_RadialPlot
Aliquot Size	*aliquotsize*	Luminescence::calc_AliquotSize
Dose Recovery Test	*doserecovery*	Luminescence::plot_DRTResults
Cosmic Dose Rate	*cosmicdose*	Luminescence::calc_CosmicDoseRate
CW Curve Transformation	*transformCW*	Luminescence::convert_CW2pHMi, Luminescence::convert_CW2
Filter Combinations	*filter*	Luminescence::plot_FilterCombinations
Fast Ratio	*fastratio*	Luminescence::calc_FastRatio
Fading Correction	*fading*	Luminescence::analyse_FadingMeasurement, Luminescence::calc
Finite Mixture	*finitemixture*	Luminescence::calc_FiniteMixture
Huntley (2006)	*huntley2006*	Luminescence::calc_Huntley2006
IRSAR RF	*irsarRF*	Luminescence::analyse_IRSAR.RF
LM Curve	*lmcurve*	Luminescence::fit_LMCurve
Test Stimulation Power	*teststimulationpower*	Luminescence::plot_RLum
Scale Gamma Dose Rate	*scalegamma*	Luminescence::scale_GammaDose
RCarb app	*RCarb*	RCarb::model_DoseRate

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

Usage

```
app_RLum(app = NULL, ...)
```

Arguments

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

How to cite

Burow, C., 2025. app_RLum(): Run Luminescence shiny apps. In: Burow, C., Wolpert, U.T., Kreutzer, S., Colombo, M., 2025. RLumShiny: 'Shiny' Applications for the R Package 'Luminescence'. R package version 0.2.5. https://tzerk.github.io/RLumShiny/

Author(s)

Christoph Burow, University of Cologne (Germany) , RLum Developer Team

See Also

shiny::runApp

Examples

```
## Not run:
# Dashboard
app_RLum()

# Plotting apps
app_RLum("abanico")
app_RLum("histogram")
app_RLum("KDE")
app_RLum("radialplot")
app_RLum("doserecovery")

# Further apps
app_RLum("aliquotsize")
app_RLum("cosmicdose")
app_RLum("transformCW")
app_RLum("filter")
app_RLum("fastratio")
app_RLum("fading")
app_RLum("finitemixture")
app_RLum("huntley2006")
app_RLum("irsarRF")
app_RLum("lmcurve")
app_RLum("surfaceexposure")
app_RLum("teststimulationpower")
app_RLum("scalegamma")
app_RLum("RCarb")

## 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 (*optional*): Display label for the control, or NULL for no label.
value	character (*optional*): Initial RGB value of the color picker. Default is black ('#000000').
position	character (*with default*): Position of the picker relative to the text input ('bottom', 'left', 'top', 'right').
color	character (*with default*): Picker color scheme ('transparent' by default). Use RGB color coding ('000000').
mode	character (*with default*): Mode of hue, saturation and value. Can either be 'HSV' or 'HVS'.
slider	logical (*with default*): Show or hide the slider.
close	logical (*with default*): Show or hide a close button.

See Also

Other input.elements: shiny::animationOptions, shiny::sliderInput; shiny::checkboxGroupInput; shiny::checkboxInput; shiny::dateInput; shiny::dateRangeInput; shiny::fileInput; shiny::numericInput; shiny::passwordInput; shiny::radioButtons; shiny::selectInput, shiny::selectizeInput; shiny::submitButton; shiny::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 (**required**): Text to be displayed in the popover.
header	character (*optional*): Optional header in the popover.
html	logical (*with default*): Insert HTML into the popover.
class	logical (*with default*): Bootstrap button class (e.g. "btn btn-danger").
placement	character (*with default*): 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 (*with default*): 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)
```

`RLumShinyAddin` *RLumShiny Dashboard Addin*

Description

RLumShiny dashboard

Usage

```
RLumShinyAddin()
```

`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` (**required**): Text to be displayed in the tooltip.
`attr`	`character` (*optional*): Attach tooltip to all elements with attribute `attr='refId'`.
`animation`	`logical` (*with default*): Apply a CSS fade transition to the tooltip.
`delay`	`numeric` (*with default*): Delay showing and hiding the tooltip (ms).
`html`	`logical` (*with default*): Insert HTML into the tooltip.
`placement`	`character` (*with default*): 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` (*with default*): 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