

Package ‘RLumShiny’

July 12, 2025

Type Package

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

Version 0.2.5

Date 2025-07-12

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 | file LICENSE

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

NeedsCompilation no

Author Christoph Burow [aut, cre] (ORCID:

<<https://orcid.org/0000-0002-5023-4046>>),

Urs Tilmann Wolpert [aut],

Sebastian Kreutzer [aut] (ORCID:

<<https://orcid.org/0000-0002-0734-2199>>),

Marco Colombo [aut] (ORCID: <<https://orcid.org/0000-0001-6672-0623>>),

R Luminescence Package Team [ctb],

Jan Odvarko [cph] (jscolor.js in www/jscolor),

AnalytixWare [cph] (ShinySky package),

RStudio [cph] (chooser_inputBinding.js in www/ and chooser.R in R/)

Maintainer Christoph Burow <christoph.burow@gmx.net>
Repository CRAN
Date/Publication 2025-07-12 18:10:02 UTC

Contents

RLumShiny-package	2
app_RLum	3
jscolorInput	5
popover	6
RLumShinyAddin	7
tooltip	8
Index	10

RLumShiny-package	<i>Shiny Applications for the R Package Luminescence</i>
-------------------	--

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	<i>abanico</i>	<code>Luminescence::plot_AbanicoPlot</code>
Histogram	<i>histogram</i>	<code>Luminescence::plot_Histogram</code>
Kernel Density Estimate Plot	<i>KDE</i>	<code>Luminescence::plot_KDE</code>
Radial Plot	<i>radialplot</i>	<code>Luminescence::plot_RadialPlot</code>
Aliquot Size	<i>aliquotsize</i>	<code>Luminescence::calc_AliquotSize</code>
Dose Recovery Test	<i>doserecovery</i>	<code>Luminescence::plot_DRTRResults</code>
Cosmic Dose Rate	<i>cosmicdose</i>	<code>Luminescence::calc_CosmicDoseRate</code>
CW Curve Transformation	<i>transformCW</i>	<code>Luminescence::convert_CW2pHMi</code> , <code>Luminescence::convert_CW2pL</code>
Filter Combinations	<i>filter</i>	<code>Luminescence::plot_FilterCombinations</code>
Fast Ratio	<i>fastratio</i>	<code>Luminescence::calc_FastRatio</code>
Fading Correction	<i>fading</i>	<code>Luminescence::analyse_FadingMeasurement</code> , <code>Luminescence::calc_Fa</code>
Finite Mixture	<i>finitemixture</i>	<code>Luminescence::calc_FiniteMixture</code>
Huntley (2006)	<i>huntley2006</i>	<code>Luminescence::calc_Huntley2006</code>
IRSAR RF	<i>irsarRF</i>	<code>Luminescence::analyse_IRSAR.RF</code>
LM Curve	<i>lmcurve</i>	<code>Luminescence::fit_LMCurve</code>
Test Stimulation Power	<i>teststimulationpower</i>	<code>Luminescence::plot_RLum</code>
Scale Gamma Dose Rate	<i>scalegamma</i>	<code>Luminescence::scale_GammaDose</code>
RCarb app	<i>RCarb</i>	<code>RCarb::model_DoseRate</code>

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

Author(s)

Christoph Burow, University of Cologne (Germany)

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

<code>inputId</code>	character (required) : Specifies the input slot that will be used to access the value.
<code>label</code>	character (optional) : Display label for the control, or NULL for no label.
<code>value</code>	character (optional) : Initial RGB value of the color picker. Default is black ('#000000').
<code>position</code>	character (with default) : Position of the picker relative to the text input ('bottom', 'left', 'top', 'right').
<code>color</code>	character (with default) : Picker color scheme ('transparent' by default). Use RGB color coding ('000000').
<code>mode</code>	character (with default) : Mode of hue, saturation and value. Can either be 'HSV' or 'HVS'.
<code>slider</code>	logical (with default) : Show or hide the slider.
<code>close</code>	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 (<i>with default</i>): 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 (<i>with default</i>): 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

app_RLum, [2](#), [3](#)

character, [4–8](#)

jscolorInput, [2](#), [5](#)

logical, [5](#), [6](#), [8](#)

Luminescence::analyse_FadingMeasurement,
[3](#)

Luminescence::analyse_IRSAR.RF, [3](#)

Luminescence::calc_AliquotSize, [3](#)

Luminescence::calc_CosmicDoseRate, [3](#)

Luminescence::calc_FadingCorr, [3](#)

Luminescence::calc_FastRatio, [3](#)

Luminescence::calc_FiniteMixture, [3](#)

Luminescence::calc_Huntley2006, [3](#)

Luminescence::convert_CW2pHMi, [3](#)

Luminescence::convert_CW2pLM, [3](#)

Luminescence::convert_CW2pLMi, [3](#)

Luminescence::convert_CW2pPMi, [3](#)

Luminescence::fit_LMCurve, [3](#)

Luminescence::plot_AbanicoPlot, [3](#)

Luminescence::plot_DRTResults, [3](#)

Luminescence::plot_FilterCombinations,
[3](#)

Luminescence::plot_Histogram, [3](#)

Luminescence::plot_KDE, [3](#)

Luminescence::plot_RadialPlot, [3](#)

Luminescence::plot_RLum, [3](#)

Luminescence::scale_GammaDose, [3](#)

numeric, [8](#)

popover, [2](#), [6](#)

RCarb::model_DoseRate, [3](#)

RLumShiny (RLumShiny-package), [2](#)

RLumShiny-package, [2](#)

RLumShinyAddin, [7](#)

shiny::animationOptions, [5](#)

shiny::checkboxGroupInput, [5](#)

shiny::checkboxInput, [5](#)

shiny::dateInput, [5](#)

shiny::dateRangeInput, [5](#)

shiny::fileInput, [5](#)

shiny::numericInput, [5](#)

shiny::passwordInput, [5](#)

shiny::radioButtons, [5](#)

shiny::runApp, [3](#), [4](#)

shiny::selectInput, [5](#)

shiny::selectizeInput, [5](#)

shiny::sliderInput, [5](#)

shiny::submitButton, [5](#)

shiny::textInput, [5](#)

tooltip, [2](#), [8](#)