Package: colourpicker (via r-universe)

February 14, 2025

Title A Colour Picker Tool for Shiny and for Selecting Colours in Plots

Version 1.3.0

Description A colour picker that can be used as an input in 'Shiny' apps or Rmarkdown documents. The colour picker supports alpha opacity, custom colour palettes, and many more options. A Plot Colour Helper tool is available as an 'RStudio' Addin, which helps you pick colours to use in your plots. A more generic Colour Picker 'RStudio' Addin is also provided to let you select colours to use in your R code.

URL https://github.com/daattali/colourpicker, https://daattali.com/shiny/colourInput/

BugReports https://github.com/daattali/colourpicker/issues

Depends R (>= 3.1.0)

Imports ggplot2, htmltools, htmlwidgets (>= 0.7), jsonlite, miniUI (>= 0.1.1), shiny (>= 0.11.1), shinyjs (>= 2.0.0), utils

Suggests knitr (>= 1.7), rmarkdown, rstudioapi (>= 0.5), shinydisconnect

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Config/pak/sysreqs make zlib1g-dev

Repository https://daattali.r-universe.dev

RemoteUrl https://github.com/daattali/colourpicker

RemoteRef HEAD

RemoteSha b223a104b36f5268902853227b50162746623a07

Contents

`colourInput` *Create a colour input control*

Description

Create an input control to select a colour.

Usage

```
colourInput(
  inputId,
  label,
  value = "white",
  showColour = c("both", "text", "background"),
  palette = c("square", "limited"),
  allowedCols = NULL,
  allowTransparent = FALSE,
  returnName = FALSE,
  closeOnClick = FALSE,
  width = NULL
)
```

Arguments

`inputId`	The `input` slot that will be used to access the value.
`label`	Display label for the control, or 'NULL for no label.
`value`	Initial value (can be a colour name or HEX code)
`showColour`	Whether to show the chosen colour as text inside the input, as the background colour of the input, or both (default).
`palette`	The type of colour palette to allow the user to select colours from. `square` (default) shows a square colour palette that allows the user to choose any colour, while `limited` only gives the user a predefined list of colours to choose from.
`allowedCols`	A list of colours that the user can choose from. Only applicable when `palette == "limited"`. The `limited` palette uses a default list of 40 colours if `allowedCols` is not defined. If the colour specified in `value` is not in the list, the default colour will revert to black.

`allowTransparent`
: If `TRUE`, enables a slider to choose an alpha (transparency) value for the colour. When a colour with opacity is chosen, the return value is an 8-digit HEX code.

`returnName`
: If `TRUE`, then return the name of an R colour instead of a HEX value when possible.

`closeOnClick`
: If `TRUE`, then the colour selection panel will close immediately after selecting a colour.

`width`
: The width of the input, e.g. `"400px"` or `"100%"`

Details

A colour input allows users to select a colour by clicking on the desired colour, or by entering a valid colour in the input box. Colours can be specified as either names ("blue"), HEX codes ("#0000FF"), RGB codes ("rgb(0, 0, 255)"), or HSL codes ("hsl(240, 100, 50)"). Use `allowTransparent = TRUE` to allow selecting semi-transparent colours. The return value is a HEX value by default, but you can use the `returnName = TRUE` parameter to get an R colour name instead (only when an R colour exists for the selected colour).

When `allowTransparent = TRUE`, the user can type into the input field any RGBA value, HSLA value, or 8-digit HEX with alpha channel You can also use any of these values as the `value` argument as the initial value of the input.

Note

See https://daattali.com/shiny/colourInput/ for a live demo.

See Also

`updateColourInput` `colourPicker`

Examples

```
if (interactive()) {
  # Example 1
  library(shiny)
  shinyApp(
    ui = fluidPage(
      colourInput("col", "Choose colour", "red"),
      plotOutput("plot")
    ),
    server = function(input, output, session) {
      output$plot <- renderPlot({
        plot(1:10, col = input$col)
      })
    }
  )

  # Example 2
  library(shiny)
  shinyApp(
    ui = fluidPage(
```

```
      strong("Selected colour:", textOutput("value", inline = TRUE)),
      colourInput("col", "Choose colour", "red"),
      h3("Update colour input"),
      textInput("text", "New colour: (colour name or HEX value)"),
      selectInput("showColour", "Show colour",
        c("both", "text", "background")),
      selectInput("palette", "Colour palette",
        c("square", "limited")),
      checkboxInput("allowTransparent", "Allow transparent", FALSE),
      checkboxInput("returnName", "Return R colour name", FALSE),
      actionButton("btn", "Update")
    ),
    server = function(input, output, session) {
      observeEvent(input$btn, {
        updateColourInput(session, "col",
          value = input$text, showColour = input$showColour,
          allowTransparent = input$allowTransparent,
          palette = input$palette,
          returnName = input$returnName)
      })
      output$value <- renderText(input$col)
    }
  )
}
```

`colourPicker` *Colour picker gadget*

Description

This gadget lets you choose colours easily. You can select multiple colours, and you can either choose any RGB colour, or browse through R colours.

Usage

```
colourPicker(numCols = 3)
```

Arguments

`numCols`	The number of colours to select when the gadget launches (you can add and remove more colours from the app itself too)

Value

Vector of selected colours

Note

This gadget returns a vector of colours that can be assigned to a variable. If instead you want to get a text representation of the colours that can embedded into code, use the addin from the RStudio Addins menu.

Examples

```
if (interactive()) {
  cols <- colourPicker(5)
}
```

`colourWidget` *Create a colour picker htmlwidget*

Description

Create a colour picker htmlwidget. This is not terribly useful right now since you can use the more powerful `colourInput` in Shiny apps and Rmarkdown documents, but this gives you an htmlwidget version of that colour picker.

Usage

```
colourWidget(
  value = "white",
  showColour = c("both", "text", "background"),
  palette = c("square", "limited"),
  allowedCols = NULL,
  allowTransparent = FALSE,
  returnName = FALSE,
  closeOnClick = FALSE,
  width = "300px",
  height = "35px",
  elementId = NULL
)
```

Arguments

`value`	Initial value (can be a colour name or HEX code)
`showColour`	Whether to show the chosen colour as text inside the input, as the background colour of the input, or both (default).
`palette`	The type of colour palette to allow the user to select colours from. `square` (default) shows a square colour palette that allows the user to choose any colour, while `limited` only gives the user a predefined list of colours to choose from.
`allowedCols`	A list of colours that the user can choose from. Only applicable when `palette == "limited"`. The `limited` palette uses a default list of 40 colours if `allowedCols` is not defined. If the colour specified in `value` is not in the list, the default colour will revert to black.
`allowTransparent`	If `TRUE`, enables a slider to choose an alpha (transparency) value for the colour. When a colour with opacity is chosen, the return value is an 8-digit HEX code.
`returnName`	If `TRUE`, then return the name of an R colour instead of a HEX value when possible.

closeOnClick	If TRUE, then the colour selection panel will close immediately after selecting a colour.
width	Custom width for the input field.
height	Custom height for the input field.
elementId	Use an explicit element ID for the widget (rather than an automatically generated one).

Examples

```
colourWidget()
colourWidget("red", palette = "limited", allowedCols = c("yellow", "red", "#123ABC"))
```

---

plotHelper *Plot colour helper*

Description

Allows you to interactively pick combinations of colours, to help you choose colours to use in your plots. The plot updates in real-time as you pick colours.

If you often find yourself spending a lot of time re-creating the same plot over and over with different colours to try to find the best colours, then the Plot Colour Helper can help you immensely.

Important: The colours you pick will be available as a variable called CPCOLS, so you can use CPCOLS in your plot code. See the example below.

Usage

```
plotHelper(code = "", colours = NULL, returnCode = FALSE)
```

Arguments

code	Code for a plot. You can use the variable CPCOLS in this code to refer to the colours that you will pick. If you do not provide any code, the plot helper will initialize with sample code. The code can be provided as text or as R code.
colours	A vector of colours to use as the initial colours in the tool, or an integer. If an integer is provided instead of colours, the tool will load with that number of colours, and default colours will be used initially. If you do not provide this parameter, the tool will attempt to guess how many colours are needed in the code and initialize that many colours.
returnCode	If TRUE, return the plot code and the CPCOLS variable as text. If FALSE (default), return the vector of selected colours.

Details

There are many keyboard shortcuts to help you be more efficient. For example, pressing *spacebar* adds a new colour, *left*/*right* keys let you navigate between the selected colours, *1-9* let you select any of the first 9 colours. For a full list of keyboard shortcuts, click on *Show keyboard shortcuts*.

Value

When this function is called using `plotHelper()`, the chosen colours are returned as a vector of colours. When this is run as an RStudio addin (through the *Addins* menu), the resulting code that includes the colour vector gets inserted into the R document. As a side effect, `CPCOLS` gets assigned in the global environment to the value of the selected colours.

Examples

```
if (interactive()) {
  cols <- plotHelper()
  cols <- plotHelper(colours = c("red", "blue"))
  cols <- plotHelper(colours = 5)

  library(ggplot2)
  cols <- plotHelper(ggplot(mtcars, aes(mpg,wt)) +
                     geom_point(aes(col = as.factor(cyl)))+
                     scale_colour_manual(values = CPCOLS))
}
```

`runExample` *Run a colourpicker example*

Description

Launch a `colourpicker` example Shiny app that shows how to use the `colourInput` control. The example is also available online.

Usage

```
runExample()
```

Examples

```
## Only run this example in interactive R sessions
if (interactive()) {
  runExample()
}
```

`updateColourInput` *Change the value of a colour input*

Description

Change the value of a colour input on the client.

Usage

```
updateColourInput(
  session,
  inputId,
  label = NULL,
  value = NULL,
  showColour = NULL,
  palette = NULL,
  allowedCols = NULL,
  allowTransparent = NULL,
  returnName = NULL,
  closeOnClick = NULL
)
```

Arguments

`session`	The `session` object passed to function given to `shinyServer`.
`inputId`	The id of the colour input object.
`label`	The label to set for the input object.
`value`	The value to set for the input object.
`showColour`	Whether to show the chosen colour as text inside the input, as the background colour of the input, or both (default).
`palette`	The type of colour palette to allow the user to select colours from. `square` (default) shows a square colour palette that allows the user to choose any colour, while `limited` only gives the user a predefined list of colours to choose from.
`allowedCols`	A list of colours that the user can choose from. Only applicable when `palette == "limited"`. The `limited` palette uses a default list of 40 colours if `allowedCols` is not defined. If the colour specified in `value` is not in the list, the default colour will revert to black.
`allowTransparent`	If `TRUE`, enables a slider to choose an alpha (transparency) value for the colour. When a colour with opacity is chosen, the return value is an 8-digit HEX code.
`returnName`	If `TRUE`, then return the name of an R colour instead of a HEX value when possible.
`closeOnClick`	If `TRUE`, then the colour selection panel will close immediately after selecting a colour.

Details

The update function sends a message to the client, telling it to change the settings of a colour input object.
This function works similarly to the update functions provided by shiny.
Any argument with NULL values will be ignored.

Note

See https://daattali.com/shiny/colourInput/ for a live demo.

See Also

colourInput

Examples

```
if (interactive()) {
  library(shiny)
  shinyApp(
    ui = fluidPage(
      div("Selected colour:", textOutput("value", inline = TRUE)),
      colourInput("col", "Choose colour", "red"),
      h3("Update colour input"),
      textInput("text", "New colour: (colour name or HEX value)"),
      selectInput("showColour", "Show colour",
        c("both", "text", "background")),
      checkboxInput("allowTransparent", "Allow transparent", FALSE),
      checkboxInput("returnName", "Return R colour name", FALSE),
      actionButton("btn", "Update")
    ),
    server = function(input, output, session) {
      observeEvent(input$btn, {
        updateColourInput(session, "col",
          value = input$text, showColour = input$showColour,
          allowTransparent = input$allowTransparent,
          returnName = input$returnName)
      })
      output$value <- renderText(input$col)
    }
  )
}
```

Index