Package 'picker'

May 9, 2026

Title Pick Data Points from a Deck.gl Scatterplot

Version 0.2.6

Description Performant interactive scatterplot for ~ 1 million points. Zoom, pan, and pick points. Includes tooltips, labels, a grid overlay, legend, and coupled interactions across multiple plots.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.2

URL https://github.com/hms-dbmi/picker

BugReports https://github.com/hms-dbmi/picker/issues

Imports htmlwidgets (>= 1.5.3)

Suggests shiny, scales

Depends R (>= 4.0)

NeedsCompilation no

Author Alex Pickering [aut, cre]

Maintainer Alex Pickering <alexvpickering@gmail.com>

Repository CRAN

Date/Publication 2022-03-31 07:30:07 UTC

Contents

picker *Render a Picker Widget*

Description

Render a Picker Widget

Usage

```
picker(
  coords,
  colors,
  labels,
  title = NULL,
  label_coords = NULL,
  polygons = NULL,
  point_color_polygons = NULL,
  show_controls = TRUE,
  grid_legend_items = NULL,
  scale_legend_props = NULL,
  scatter_props = NULL,
  deck_props = NULL,
  text_props = NULL,
  polygon_props = NULL,
  xrange = NULL,
  yrange = NULL,
  xaxs = 0.04,
  yaxs = 0.04,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

coords	data.frame with two columns. First has x, second has y coordinates.
colors	vector of hex colors, one for each row of coords.
labels	vector of point labels used for tooltips on hover.
title	character string to show in top left of plot.
label_coords	data.frame with three columns 'x', 'y', and 'label'. Used for text layer.
polygons	data.frame containing at minimum columns 'x1', 'x2', 'y1', 'y2', that define the polygons to draw and 'color' that defines the color.
point_color_polygons	character, a color to make points when polygons are shown e.g. 'white'.
show_controls	Should control panel be shown? Default is TRUE.

`grid_legend_items`
: list of lists with `color` hex for legend square and `label` for legend items. Only visible for grid display.

`scale_legend_props`
: optional props to render a gradient scale legend. For example: `list(colorHigh = 'red', colorLow = 'gray', high = 4, low = 0)`

`scatter_props` Props passed to deck.gl ScatterplotLayer.

`deck_props` Props passed to deck.gl Deck instance.

`text_props` Props passed to deck.gl TextLayer.

`polygon_props` Props passed to deck.gl PolygonLayer.

`xrange` range of x-values. Default is `range(coords[,1])`.

`yrange` range of y-values. Default is `range(coords[,2])`.

`xaxs` the fraction to extend `xrange` on either side. Default is 0.04.

`yaxs` the fraction to extend `yrange` on either side. Default is 0.04.

`width` width of htmlwidget.

`height` height of htmlwidget.

`elementId` id of htmlwidget.

Value

renders html widget

Examples

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

  # load example data
  load(system.file('extdata/pbmcs.rda', package = 'picker'))

  # setup gradient scale legend
  scale_legend_props <- list(
    colorHigh = 'blue',
    colorLow = '#f5f5f5',
    high = round(max(exp)),
    low = min(exp))

  text_props <- list()

  # get colors for gene expression
  exp <- scales::rescale(exp, c(0, 1))
  expression_colors <- scales::seq_gradient_pal('#f5f5f5', 'blue')(exp)

  # legend to show when grid is visible
  grid_legend_items = list(
    list(color = '#FF0000', label = '\U2191'),
```

```
  list(color = '#0000FF', label = '\U2193'),
  list(color = '#989898', label = 'p \U003C .05'),
  list(color = '#EAEAEA', label = 'p \U2265 .05')
)

ui = shinyUI(fluidPage(
  tags$head(tags$style(".picker {border: 1px solid #ddd; margin: 20px 0;}")),
  shiny::column(
    width = 6,
    pickerOutput('clusters', width = '100%', height = '400px'),
    pickerOutput('expression', width = '100%', height = '400px'),
    verbatimTextOutput('selected')
  )
))

server = function(input, output) {

  # show selected output
  output$selected <- renderPrint({
    input$clusters_selected_points
  })

  # coordinate views (zoom/pan)
  clusters_proxy <- picker_proxy('clusters')
  observeEvent(input$expression_view_state, {
    update_picker(clusters_proxy, input$expression_view_state)
  })

  expression_proxy <- picker_proxy('expression')
  observeEvent(input$clusters_view_state, {
    update_picker(expression_proxy, input$clusters_view_state)
  })

  # change title between grid/scatterplot
  observeEvent(input$clusters_show_grid, {
    title <- ifelse(input$clusters_show_grid, '\U0394 CELLS', '')
    update_picker(clusters_proxy, title = title)
  })


  # render pickers
  output$clusters <- renderPicker(
    picker(
      coords,
      cluster_colors,
      labels,
      label_coords = label_coords,
      polygons = polygons,
      text_props = text_props,
      point_color_polygons = 'white',
      grid_legend_items = grid_legend_items)
  )
```

```
    output$expression <- renderPicker(
      picker(coords,
             expression_colors,
             labels,
             show_controls = FALSE,
             scale_legend_props = scale_legend_props)
    )
  }

  shinyApp(ui = ui, server = server, options = list(launch.browser = TRUE))
}
```

`picker-shiny` *Shiny bindings for picker*

Description

Output and render functions for using picker within Shiny applications and interactive Rmd documents.

Usage

```
pickerOutput(outputId, width = "100%", height = "400px")

renderPicker(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	output variable to read from
`width, height`	Must be a valid CSS unit (like `'100%'`, `'400px'`, `'auto'`) or a number, which will be coerced to a string and have `'px'` appended.
`expr`	An expression that generates a picker
`env`	The environment in which to evaluate `expr`.
`quoted`	Is `expr` a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

Value

An output or render function that enables the use of the widget within Shiny applications.

picker_proxy *Create a picker proxy object*

Description

Creates a picker-like object that can be used to update a picker object that has already been rendered.

Usage

```
picker_proxy(shinyId, session = shiny::getDefaultReactiveDomain())
```

Arguments

shinyId	single-element character vector indicating the output ID of the deck to modify
session	the Shiny session object to which the picker widget belongs; usually the default value will suffice.

Value

a picker_proxy object that can be updated with update_picker.

See Also

update_picker

update_picker *Send commands to a picker instance in a* Shiny *app*

Description

Send commands to a picker instance in a Shiny app

Usage

```
update_picker(
  proxy,
  view_state = NULL,
  colors = NULL,
  labels = NULL,
  label_coords = NULL,
  polygons = NULL,
  show_grid = NULL,
  title = NULL
)
```

Arguments

proxy	picker proxy object created by picker_proxy.
view_state	view state from other picker input (optional).
colors	vector of hex colors, one for each row of coords.
labels	vector of point labels used for tooltips on hover.
label_coords	data.frame with three columns 'x', 'y', and 'label'. Used for text layer.
polygons	data.frame containing at minimum columns 'x1', 'x2', 'y1', 'y2', that define the polygons to draw and 'color' that defines the color.
show_grid	set to TRUE to turn on grid layer.
title	character string to show in top left of plot.

Value

The original proxy object. Called for side effects.

See Also

picker_proxy

Index