# Package ‘vfinputs’

October 13, 2020

**Type**  Package  
**Title**  Visual Filter Inputs for Shiny  
**Version**  0.1.0  
**Date**  2020-09-28  
**Depends**  R (>= 3.0.2)  
**Imports**  shiny, htmltools, jsonlite, scales  
**Suggests**  RColorBrewer, testthat  
**Maintainer**  Rafael Henkin &lt;r.henkin@qmul.ac.uk&gt;  
**Description**  A set of visual input controls for Shiny apps to facilitate filtering across multiple outputs.  
**License**  GPL-3  
**LazyData**  TRUE  
**RoxygenNote**  7.1.1  
**Encoding**  UTF-8  
**URL**  https://github.com/rhenkin/vfinputs  
**BugReports**  https://github.com/rhenkin/vfinputs/issues  
**NeedsCompilation**  no  
**Author**  Rafael Henkin [cre, aut] (<https://orcid.org/0000-0002-5511-5230>), Mike Bostock [cph] (D3.js library, https://d3js.org)  
**Repository**  CRAN  
**Date/Publication**  2020-10-13 15:20:03 UTC  

## R topics documented:

- addCategoryBlocks ........................................... 2  
- addColorStrips ........................................... 3  
- categoricalColorFilter ........................................... 3  
- categoricalLegend ........................................... 5  
- categoryBlock ........................................... 6  
- colorStrip ........................................... 7
addCategoryBlocks

Add list of category items

Description

Add list of category items

Usage

addCategoryBlocks(orient, input_id, color_map, values)

Arguments

orient  Orientation of the legend. Can be "bottom" (default, horizontal with labels below), "top" (horizontal with labels above), "left" (vertical with labels on the left) and "right" (vertical with labels on the right).

input_id  The CSS class used to trigger interactions

color_map  A list of colors from where the corresponding item color will be retrieved

values  A list of values from which the corresponding item label will be retrieved

Value

A list of the same length as values, containing either "span" or "div" elements depending on the chosen orientation.

See Also

categoricalLegend()
**addColorStrips**

Add list of colored strips

**Description**

Add list of colored strips

**Usage**

```r
addColorStrips(n_strips, color_map, orient, pos_function, size, thickness = 20)
```

**Arguments**

- `n_strips`: Number of strips to be added
- `color_map`: A list of colors corresponding to the number of strips
- `orient`: Orientation of the legend. Can be "bottom" (default, horizontal with labels below), "top" (horizontal with labels above), "left" (vertical with labels on the left) and "right" (vertical with labels on the right).
- `pos_function`: A function to convert from index number to pixels
- `size`: The length of the list in pixels
- `thickness`: The height or width of the list in pixels

**Value**

A list of SVG `rect` shapes.

**See Also**

- `numericLegend()`

---

**categoricalColorFilter**

Add a visual filter input for categorical data

**Description**

Add a visual filter input for categorical data

**Usage**

```r
categoricalColorFilter(inputId, ...)
```
Arguments

inputId The input slot that will be used to access the value.

... Arguments passed on to categoricalLegend

label Display label for the control, or NULL for no label.
class The CSS class of the input div element to match with any filter toggling functions. Default class is "categorical-color-filter".

values List of character vectors that will match with the colors or palette in the order provided by both.

data Alternative vector to extract values with "unique()" function.

colors Colours to match with values; must be a valid argument to grDevices::col2rgb(). This can be a character vector of "#RRGGBB" or "#RRGGBBAA", colour names from grDevices::colors(), or a positive integer that indexes into grDevices::palette().

palette A function that outputs a list of colors.

orient Orientation of the legend. Can be "bottom" (default, horizontal with labels below), "top" (horizontal with labels above), "left" (vertical with labels on the left) and "right" (vertical with labels on the right).

size Absolute length in pixels of the color bar; becomes width or height depending on value of orient. Default is 220.

multiple Is selection of multiple items allowed? Default is TRUE. With FALSE, selecting one item will de-select the others.

Value

A visual filter input control that can be added to a UI definition

Server value

start and end bounds of a selection. The default value (or empty selection) is NULL.

See Also

categoricalLegend()

Other visual filters: continuousColorFilter(), discreteColorFilter()

Examples

## Only run examples in interactive R sessions
if (interactive()) {

ui <- fluidPage(
    categoricalColorFilter("filter", data = sort(mtcars$gear), orient = "right",
                            palette = RColorBrewer::brewer.pal(8, "Dark2")),
    verbatimTextOutput("value")
)
server <- function(input, output) {
    output$value <- output$selection <- renderPrint({
        if (!is.null(input$filter)) {
            format(input$filter)
        }
    })
}
categoricalLegend

Create a categorical legend

Description

Create a color legend based on given data and palette or colors. Also passes on data- attributes for optional JS interaction.

Usage

categoricalLegend(
  inputId, 
  label = NULL, 
  class = "", 
  values = NULL, 
  data = NULL, 
  colors = NULL, 
  palette = NULL, 
  orient = "bottom", 
  size = 220, 
  multiple = TRUE
)
Arguments

inputId  The input slot that will be used to access the value.

label  Display label for the control, or NULL for no label.

class  The CSS class of the input div element to match with any filter toggling functions. Default class is "categorical-color-filter".

values  List of character vectors that will match with the colors or palette in the order provided by both.

data  Alternative vector to extract values with "unique()" function.

colors  Colours to match with values; must be a valid argument to `grDevices::col2rgb()`. This can be a character vector of "#RRGGBB" or "#RRGGBBAA", colour names from `grDevices::colors()`, or a positive integer that indexes into `grDevices::palette()`.

palette  A function that outputs a list of colors.

orient  Orientation of the legend. Can be "bottom" (default, horizontal with labels below), "top" (horizontal with labels above), "left" (vertical with labels on the left) and "right" (vertical with labels on the right).

size  Absolute length in pixels of the color bar; becomes width or height depending on value of orient. Default is 220.

multiple  Is selection of multiple items allowed? Default is TRUE. With FALSE, selecting one item will de-select the others.

Value

A categorical color legend control that can be added to a UI definition

See Also

`discreteColorFilter()` `continuousColorFilter()` `categoricalColorFilter()`

Other base legend: `numericLegend()`

categoryBlock(i, values, tag_name, class, color_map)
colorStrip

Arguments

i The index of the item to be created
values A list of values from which the corresponding item label will be retrieved
tag_name An HTML element tag
class The HTML element class that will enable interaction
color_map A list of colors from where the corresponding item color will be retrieved

Value

An HTML element with pointer cursor, a colored square and a label

colorStrip Add color strip

Description

Add color strip

Usage

colorStrip(color, x = 0, y = 0, width = 1, height = 30)

Arguments

color A valid CSS color name
x The x position of the rect shape relative to a container
y The y position of the rect shape relative to a container
width The width of the rect
height The height of the rect

Value

A rect element with the color argument as fill and stroke
**Description**

The brush used in this filter allows a free selection over the whole input range.

**Usage**

```r
continuousColorFilter(inputId, ...)
```

**Arguments**

- `inputId`: The input slot that will be used to access the value.
- `...`: Arguments passed on to `numericLegend`
- `label`: Display label for the control, or `NULL` for no label.
- `class`: The CSS class of the input div element to match with any brush-defining functions. Default classes for brushes are either "continuous-color-filter" or "discrete-color-filter".
- `n`: Number of color strips in the legend. Default is 100.
- `minValue`: Minimum numeric value in the legend (can be higher the maximum for inverted scale).
- `maxValue`: Maximum numeric value in the legend (can be lower the minimum for inverted scale).
- `data`: Alternative vector to extract numeric minimum and maximum values.
- `colors`: Colours to interpolate; must be a valid argument to `grDevices::col2rgb()`.
  This can be a character vector of "#RRGGBB" or "#RRGGBBAA", colour names from `grDevices::colors()`, or a positive integer that indexes into `grDevices::palette()`.
- `palette`: A function that outputs a list of colors.
- `options`: Configuration options for brush and scale. Use `ticks` to specify number of ticks or a list of specific tick values, `format` to a d3-format-compatible formatting string (see `https://github.com/d3/d3-format` for valid formats) and `hide_brush_labels` as `TRUE` to hide the brush interval.
- `orient`: Orientation of the legend. Can be "bottom" (default, horizontal with labels below), "top" (horizontal with labels above), "left" (vertical with labels on the left) and "right" (vertical with labels on the right).
- `size`: Absolute length in pixels of the color bar; becomes width or height depending on value of `orient`. Default is 200.
- `thickness`: Absolute thickness in pixels of the color bar; opposite of `size` depending on value of `orient`. Default is 20.
- `offset`: Left offset for scale to allow long labels. Default is 0.

**Value**

A visual filter input control that can be added to a UI definition.
Server value

Start and end bounds of a selection. The input value is NULL for empty selections.

See Also

discreteColorFilter() categoricalColorFilter()

Other visual filters: categoricalColorFilter(), discreteColorFilter()

Examples

```r
## Only run examples in interactive R sessions
if (interactive()) {
  ui <- fluidPage(
    continuousColorFilter("filter", minValue = 0, maxValue = 200, palette = scales::viridis_pal()),
    verbatimTextOutput("value")
  )
  server <- function(input, output) {
    output$value <- output$selection <- renderPrint({
      if (!is.null(input$filter)) {
        paste0(input$filter$start, ",", input$filter$end)
      }
    })
  }
  shinyApp(ui, server)
}

ui <- fluidPage(
  continuousColorFilter("filter", data = mtcars$mpg, colors = c("#FF0000", "#0000FF")),
  verbatimTextOutput("value")
)
server <- function(input, output) {
  output$value <- output$selection <- renderPrint({
    if (!is.null(input$filter)) {
      paste0(input$filter$start, ",", input$filter$end)
    }
  })
}
shinyApp(ui, server)
```

discreteColorFilter  Add a visual filter input for discrete values

Description

The brush used in this filter snaps to evenly divided steps based on the number of colors passed as argument. With minValue = 0, maxValue = 100 and n = 5, it will snap at the edges (0 and 100) and 20, 40, 60, and 80.
Usage

discreteColorFilter(inputId, ...)

Arguments

- **inputId**: The input slot that will be used to access the value.
- **...**: Arguments passed on to `numericLegend`
- **label**: Display label for the control, or NULL for no label.
- **class**: The CSS class of the input div element to match with any brush-defining functions. Default classes for brushes are either "continuous-color-filter" or "discrete-color-filter".
- **n**: Number of color strips in the legend. Default is 100.
- **minValue**: Minimum numeric value in the legend (can be higher than the maximum for inverted scale).
- **maxValue**: Maximum numeric value in the legend (can be lower than the minimum for inverted scale).
- **data**: Alternative vector to extract numeric minimum and maximum values.
- **colors**: Colours to interpolate; must be a valid argument to `grDevices::col2rgb()`. This can be a character vector of "#RRGGBB" or "#RRGGBBAA", colour names from `grDevices::colors()`, or a positive integer that indexes into `grDevices::palette()`.
- **palette**: A function that outputs a list of colors.
- **options**: Configuration options for brush and scale. Use `ticks` to specify number of ticks or a list of specific tick values, `format` to a d3-format-compatible formatting string (see [https://github.com/d3/d3-format](https://github.com/d3/d3-format) for valid formats) and `hide_brush_labels` as TRUE to hide the brush interval.
- **orient**: Orientation of the legend. Can be "bottom" (default, horizontal with labels below), "top" (horizontal with labels above), "left" (vertical with labels on the left) and "right" (vertical with labels on the right).
- **size**: Absolute length in pixels of the color bar; becomes width or height depending on value of `orient`. Default is 200.
- **thickness**: Absolute thickness in pixels of the color bar; opposite of `size` depending on value of `orient`. Default is 20.
- **offset**: Left offset for scale to allow long labels. Default is 0.

Value

A visual filter input control that can be added to a UI definition.

Server value

start and end bounds of a selection. The input value is NULL for empty selections.

start and end bounds of a selection. The default value is NULL.

See Also

- `numericLegend()`
- Other visual filters: `categoricalColorFilter()`, `continuousColorFilter()`
Examples

```r
## Only run examples in interactive R sessions
if (interactive()) {

ui <- fluidPage(
  discreteColorFilter("filter", minValue = 0, maxValue = 200, n = 5,
                       palette = scales::viridis_pal()),
  verbatimTextOutput("value")
)
server <- function(input, output) {
  output$value <- output$selection <- renderPrint({
    if (!is.null(input$filter)) {
      paste0(input$filter$start, ",", input$filter$end)
    }
  })
}  
shinyApp(ui, server)
}
```

---

**numericLegend**

Create a numeric legend

**Description**

Create a color legend based on given data and palette or colors. Also passes on data- attributes for optional JS interaction.

**Usage**

```r
numericLegend(
  inputId,  
  label = NULL,  
  class = "",  
  n = 100,  
  minValue = NULL,  
  maxValue = NULL,  
  data = NULL,  
  colors = NULL,  
  palette = NULL,  
  options = NULL,  
  orient = "bottom",  
  size = 200,  
  thickness = 20,  
  offset = 0
)
```
Arguments

inputId  The input slot that will be used to access the value.
label    Display label for the control, or NULL for no label.
class    The CSS class of the input div element to match with any brush-defining functions. Default classes for brushes are either "continuous-color-filter" or "discrete-color-filter".
n        Number of color strips in the legend. Default is 100.
minValue Minimum numeric value in the legend (can be higher the maximum for inverted scale).
maxValue  Maximum numeric value in the legend (can be lower the minimum for inverted scale).
data     Alternative vector to extract numeric minimum and maximum values.
colors   Colours to interpolate; must be a valid argument to grDevices::col2rgb(). This can be a character vector of "#RRGGBB" or "#RRGGBBAA", colour names from grDevices::colors(), or a positive integer that indexes into grDevices::palette().
palette  A function that outputs a list of colors
options  Configuration options for brush and scale. Use ticks to specify number of ticks or a list of specific tick values, format to a d3-format-compatible formatting string (see https://github.com/d3/d3-format for valid formats) and hide_brush_labels as TRUE to hide the brush interval.
orient   Orientation of the legend. Can be "bottom" (default, horizontal with labels below), "top" (horizontal with labels above), "left" (vertical with labels on the left) and "right" (vertical with labels on the right).
size     Absolute length in pixels of the color bar; becomes width or height depending on value of orient. Default is 200.
thickness Absolute thickness in pixels of the color bar; opposite of size depending on value of orient. Default is 20.
offset   Left offset for scale to allow long labels. Default is 0.

Value

A numeric color legend control that can be added to a UI definition

See Also

discreteColorFilter() continuousColorFilter() categoricalColorFilter() 
Other base legend: categoricalLegend()
updateCategoricalFilter

Change a categorical legend in the client

Description

Change a categorical legend in the client

Usage

```
updateCategoricalFilter(
  session, 
  inputId, 
  label = NULL, 
  select = NULL, 
  deselect = NULL 
)
```

Arguments

- `session`: The session object passed to function given to shinyServer.
- `inputId`: The id of the input object.
- `label`: The label to set for the input object.
- `select`: Items to be selected.
- `deselect`: Items to be deselected.

Details

This function only affects the label and the selection. Re-creating the items requires deleting and re-creating the legend using shinyjs, for example.

See Also

- `categoricalColorFilter()`

Other update functions: `updateNumericFilter()`
updateNumericFilter  Change a numeric legend filter in the client

Description

This function does not validate if a brush is already defined; updating only one of start or end with
an empty brush will assign the other to NaN.

Usage

updateNumericFilter(
  session,
  inputId,
  label = NULL,
  start = NULL,
  end = NULL,
  minValue = NULL,
  maxValue = NULL
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>The session object passed to function given to shinyServer.</td>
</tr>
<tr>
<td>inputId</td>
<td>The id of the input object.</td>
</tr>
<tr>
<td>label</td>
<td>The label to set for the input object.</td>
</tr>
<tr>
<td>start</td>
<td>Beginning of selection interval.</td>
</tr>
<tr>
<td>end</td>
<td>End of selection interval.</td>
</tr>
<tr>
<td>minValue</td>
<td>Minimum numeric value in the legend (can be higher the maximum for inverted scale).</td>
</tr>
<tr>
<td>maxValue</td>
<td>Maximum numeric value in the legend (can be lower the minimum for inverted scale).</td>
</tr>
</tbody>
</table>

Details

This function only affects the label and JavaScript-implemented axis and brush values and selection.
Re-creating the color strips and changing the ticks and format of values requires deleting and re-
creating the legend using shinyjs, for example.

See Also

continuousColorFilter() discreteColorFilter()

Other update functions: updateCategoricalFilter()
Index

* **base legend**
  - categoricalLegend, 5
  - numericLegend, 11

* **update functions**
  - updateCategoricalFilter, 13
  - updateNumericFilter, 14

* **visual filters**
  - categoricalColorFilter, 3
  - continuousColorFilter, 8
  - discreteColorFilter, 9

  addCategoryBlocks, 2
  addColorStrips, 3

  categoricalColorFilter, 3, 9, 10
  categoricalColorFilter(), 6, 9, 12, 13
  categoricalLegend, 4, 5, 12
  categoricalLegend(), 2, 4
  categoryBlock, 6
  colorStrip, 7
  continuousColorFilter, 4, 8, 10
  continuousColorFilter(), 6, 12, 14

  discreteColorFilter, 4, 9, 9
  discreteColorFilter(), 6, 9, 12, 14

  grDevices::col2rgb(), 4, 6, 8, 10, 12
  grDevices::colors(), 4, 6, 8, 10, 12
  grDevices::palette(), 4, 6, 8, 10, 12

  numericLegend, 6, 8, 10, 11
  numericLegend(), 3, 10

  updateCategoricalFilter, 13, 14
  updateNumericFilter, 13, 14