# Package 'shinyfilter'

May 10, 2021

**Type** Package

**Title** Use Interdependent Filters on Table Columns in Shiny Apps

**Description** Allows to connect 'selectizeInputs' widgets as filters to a 'reactable' table. As known from spreadsheet applications, column filters are interdependent, so each filter only shows the values that are really available at the moment based on the current selection in other filters. Filter values currently not available (and also those being available) can be shown via popovers or tooltips.

**Version** 0.1.1

**Maintainer** Joachim Zuckarelli <joachim@zuckarelli.de>

**Depends** R (>= 3.5.0)

**License** GPL-3

**Imports** shiny, reactable, shinyBS, shinyjs, stringr

**Repository** CRAN

**BugReports** https://github.com/jsugarelli/shinyfilter/issues

**URL** https://github.com/jsugarelli/shinyfilter/

**Encoding** UTF-8

**ByteCompile** true

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Author** Joachim Zuckarelli [aut, cre] (<https://orcid.org/0000-0002-9280-3016>)

**Date/Publication** 2021-05-10 20:50:02 UTC

## R topics documented:

## define_filters *Define the set of interdependent filters*

### Description

Installs the filters and binds them to the `reactable` widget and the dataframe presented in the `reactable`.

`define_filters()` needs to be called in the server function of any shiny app using `shinyfilter`.

### Usage

```
define_filters(input, react_id, filters, data)
```

### Arguments

| | |
|---|---|
| `input` | The input object provided as an argument to the server function. |
| `react_id` | Object ID/input slot of the `reactable` which the filters will be linked to. |
| `filters` | A named character vector with the column names of the dataframe that will be filtered. The *names* of the vector elements are the object IDs/input slots of the respective `selectizeInput()` widgets used as filters. |
| `data` | The (unfiltered) dataframe presented in the `reactable`. |

### Details

For a full example of a shiny app using `shinyfilter` please call up the help for `update_filters()`. See the `README.md` file or the GitHub repo on https://github.com/jsugarelli/shinyfilter for a comprehensive `shinyfilter` tutorial.

### Value

No return value.

## event *Get JavaScript code for filters' selectizeInput onchange event*

### Description

Helper function to create the JavaScript event handler code for the `selectizeInput` filters of a shiny app using `shinyfilters`.

### Usage

```
event(name)
```

### Arguments

| | |
|---|---|
| name | Name of the event/input variable set by the selectizeInput filters whenever the selection changes. Can be handled in a call of observeEvent(). |

### Details

Processing the onChange event of the selectizeInput widgets that serve as the filters is necessary so that filters all the other shinyfilter filters bound to the same reactable can be updated accordingly and show the currently available filter options. All selectizeInput should have exactly the same event handler.

You can of course create the JavaScript code for the onChange event handler function yourself, especially if you want to trigger additional operations in JavaScript whenever an onChange event occurs. event() function is just intended as a shortcut to save time and effort.

For a full example of a shiny app using shinyfilter please call up the help for update_filters(). See the README.md file or the GitHub repo on https://github.com/jsugarelli/shinyfilter for a comprehensive shinyfilter tutorial.

### Value

JavaScript code for the onChange event.

### Examples

```
event("myEvent")
```

---

## update_filters *Update the filter options in each filter when the selection in any of the filters changes*

---

### Description

Updates all filters linked to a reactable. As shinyfilter filters are interdependent, update_filters() makes sure that each filter (selectizeInput widget) only shows the filter options currently available, given the selection in all other filters.

### Usage

```
update_filters(input, session, react_id)
```

### Arguments

| | |
|---|---|
| input | The input object provided as an argument to the server function. |
| session | The session variable provided as an argument to the server function. |
| react_id | The output variable/ID of the reactable for which filters will be updated. |

## Details

See below for a full example of a shiny app using `shinyfilter`. See the `README.md` file or the GitHub repo on https://github.com/jsugarelli/shinyfilter for a comprehensive `shinyfilter` tutorial.

## Value

The filtered dataframe to be presented in the `reactable` widget. Ideally, this is captured in a reactive value so that the `reactable` updates automatically.

## Examples

```
if(interactive()) {
  library(shiny)
  library(reactable)
  library(shinyfilter)

  cars_csv <- system.file("cars.csv", package="shinyfilter")

  cars <- read.csv(cars_csv, stringsAsFactors = FALSE, header = TRUE, encoding = "UTF-8")

  app = shinyApp(
    ui <- fluidPage(
      titlePanel("Cars Database"),
      sidebarLayout(
        sidebarPanel(
          width = 2,

          selectizeInput(inputId = "sel_manufacturer", label = "Manufacturer",
                         multiple = TRUE, options = list(onChange = event("ev_click")),
                         choices = sort(unique(cars$manufacturer))),

          selectizeInput(inputId = "sel_year", label = "Year",
                         multiple = TRUE, options = list(onChange = event("ev_click")),
                         choices = sort(unique(cars$year))),

          selectizeInput(inputId = "sel_fuel", label = "Fuel",
                         multiple = TRUE, options = list(onChange = event("ev_click")),
                         choices = sort(unique(cars$fuel))),

          selectizeInput(inputId = "sel_condition", label = "Condition",
                         multiple = TRUE, options = list(onChange = event("ev_click")),
                         choices = sort(unique(cars$condition))),

          selectizeInput(inputId = "sel_size", label = "Size",
                         multiple = TRUE, options = list(onChange = event("ev_click")),
                         choices = sort(unique(cars$size))),

          selectizeInput(inputId = "sel_transmission", label = "Transmission",
                         multiple = TRUE, options = list(onChange = event("ev_click")),
                         choices = sort(unique(cars$transmission))),

          selectizeInput(inputId = "sel_color", label = "Color",
```

```
                      multiple = TRUE, options = list(onChange = event("ev_click")),
                      choices = sort(unique(cars$paint_color))),

      selectizeInput(inputId = "sel_type", label = "Type",
                      multiple = TRUE, options = list(onChange = event("ev_click")),
                      choices = sort(unique(cars$type))),
      use_tooltips(background = "#'  1B3F8C", foreground = "#'  FFFFFF")
    ),
    mainPanel(
      reactableOutput(outputId = "tbl_cars")
    )
  )
),


server = function(input, output, session) {

  r <- reactiveValues(mycars = cars)

  define_filters(input,
                 "tbl_cars",
                 c(sel_manufacturer = "manufacturer",
                   sel_year = "year",
                   sel_fuel = "fuel",
                   sel_condition = "condition",
                   sel_size = "size",
                   sel_transmission = "transmission",
                   sel_color = "paint_color",
                   sel_type = "type"),
                 cars)


  observeEvent(input$ev_click, {
    r$mycars <- update_filters(input, session, "tbl_cars")
    update_tooltips("tbl_cars",
                    session,
                    tooltip = TRUE,
                    title_avail = "Available is:",
                    title_nonavail = "Currently not available is:",
                    popover_title = "My filters",
                    max_avail = 10,
                    max_nonavail = 10)
  })


  output$tbl_cars <- renderReactable({
    reactable(data = r$mycars,
              filterable = TRUE,
              rownames = FALSE,
              selection = "multiple",
              showPageSizeOptions = TRUE,
              paginationType = "jump",
              showSortable = TRUE,
```

```
                highlight = TRUE,
                resizable = TRUE,
                rowStyle = list(cursor = "pointer"),
                onClick = "select"
      )
    })

  }
 )

 runApp(app)
}
```

---

`update_tooltips` *Update the tooltips/popovers based on the currently available filter options*

---

## Description

Updates all tooltips or popovers for `shinyfilter` filter `selectizeInput` widgets. Tooltips/popovers can be used to show the currently unavailable filter options, i.e. the filter options that are not available at the moment because of the dataframe presented in the `reactable` is filtered by the choices made in the other filters. It is also possible to list the available filter options as well.

If you want to use tooltips/popovers, you need to call `use_tooltips()` from within the UI definition of your shiny app.

## Usage

```
update_tooltips(
  react_id,
  session,
  tooltip = TRUE,
  show_avail = TRUE,
  title_avail = "Available values:",
  title_nonavail = "Currently not available filters:",
  popover_title = "Filter options",
  max_avail = NULL,
  max_nonavail = max_avail,
  more_avail = "... (# more)",
  more_nonavail = "... (# more)",
  placement = "top"
)
```

## Arguments

| | |
|---|---|
| `react_id` | The output variable/ID of the `reactable` to which the filters are linked. |
| `session` | The session variable provided as an argument to the server function. |

| | |
|---|---|
| tooltip | If TRUE, tooltips will be shown. If FALSE, popovers will be shown. |
| show_avail | If TRUE not only the unavailable filter options will be listed in the tooltips/popovers, but the unavailable ones as well. |
| title_avail | Header text for the list of available filter options. |
| title_nonavail | Header text for the list of unavailable filter options. |
| popover_title | Title text for the popover window. Only relevant when tooltips = FALSE. |
| max_avail | Maximum number of available filter options shown. Use the more_avail argument to determine what is shown if the number of available filter options exceeds max_avail. |
| max_nonavail | Maximum number of non-available filter options shown. Use the more_nonavail argument to determine what is shown if the number of non-available filter options exceeds max_nonavail. |
| more_avail | Text to be shown if show_avail = TRUE and the number of available filter options exceeds max_avail. In this case, only the first max_avail filter options are shown followed by more_avail. In more_avail you can use # as a placeholder for the number of filter options exceeding max_avail. |
| more_nonavail | Text to be shown if the number of available filter options exceeds max_nonavail. In this case, only the first max_nonavail filter options are shown followed by more_nonavail. In more_nonavail you can use # as a placeholder for the number of filter options exceeding max_nonavail. |
| placement | Defines where the tooltip/popover is placed relative to the filter (i.e. selectizeInput widget) it belongs to. Can be either "top", "bottom", "left" or "right". |

## Details

For a full example of a shiny app using shinyfilter please call up the help for update_filters(). See the README.md file or the GitHub repo on https://github.com/jsugarelli/shinyfilter for a comprehensive shinyfilter tutorial.

**Tip**: If your tooltips/popovers do not show up, attach the shinyBS package directly in your Shiny app by adding library(shinyBS) to your code. The shinyBS package is used to create the tooltips and popovers.

## Value

No return value.

---

# use_tooltips *Add tooltip functionality to the app*

---

## Description

Prepares the application for the use of tooltips or popovers to show the (un)available filter options. use_tooltips() needs to be called from within the UI definition of your shiny app. See update_tooltips() for how to create the actual tooltips or popovers.

## Usage

```
use_tooltips(
  background = "#000000",
  foreground = "#FFFFFF",
  textalign = "left",
  fontsize = "100%",
  opacity = 0.8
)
```

## Arguments

| | |
|---|---|
| background | Background color of the tooltips/popovers the in CSS hex format. |
| foreground | Font color of the tooltips/popovers the in CSS hex format. |
| textalign | Alignment of the text in the tooltips/popovers; either "left", "right", "center" or "justify". |
| fontsize | Font size of the tooltips/popovers. |
| opacity | Opacity of the tooltips/popovers. |

## Details

For a full example of a shiny app using shinyfilter please call up the help for update_filters(). See the README.md file or the GitHub repo on https://github.com/jsugarelli/shinyfilter for a comprehensive shinyfilter tutorial.

## Value

No return value.

# Index