Package 'shinyfilters'

November 12, 2026

Title Create 'shiny' Inputs from Vectors, 'data.frames', or any R Object

Version 0.3.2

Description Provides an interface to 'shiny' inputs used for filtering vectors, data.frames, and other objects. 'S7'-based implementation allows for seamless extensibility.

License MIT + file LICENSE

URL https://joshwlivingston.github.io/shinyfilters/, https://github.com/joshwlivingston/shinyfilters

BugReports https://github.com/joshwlivingston/shinyfilters/issues

Imports cli, htmltools, methods, rlang, S7, shiny

Suggests bslib, DT, knitr, rmarkdown, roxygen2, shinyWidgets, testthat (>= 3.1.0), tibble

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation no

Author Josh Livingston [cre, aut], Novica Nakov [ctb] (ORCID: <https://orcid.org/0009-0005-7773-7718>)

Maintainer Josh Livingston <joshwlivingston@gmail.com>

Repository CRAN

Date/Publication 2026-11-11 23:20:02 UTC

Contents

apply_filters 2
args_filter_input 3
call_input_function 4

filterInput 5
get_filter_logical 7
get_input_ids 8
get_input_labels 9
get_input_values 9
serverFilterInput 10
updateFilterInput 12

Index **15**

`apply_filters` *Apply Filters to an object*

Description

Applies a list of filters to an object, returning the filtered object.

Usage

```
apply_filters(
  x,
  filter_list,
  filter_combine_method = "and",
  expanded = FALSE,
  cols = NULL,
  ...
)
```

Arguments

`x`	An object to filter; typically a data.frame.
`filter_list`	A named list of filter values, used to filter the values in `x`. If `filter_list` is `NULL`, `x` is returned unmodified.
`filter_combine_method`	A string or function indicating how to combine multiple filters. If a string, it can be "and" (or "&") for logical AND, or "or" (or "|") for logical OR. If a function, it should take two logical vectors and return a combined logical vector.
`expanded`	Logical; if `TRUE`, returns a named list of data.frames, each containing one column, its own, filtered according to the values of all *other* filters.
`cols`	Optional character vector of column names to retain in the output when `x` is a data.frame. If `NULL` (the default), all columns are retained.
`...`	Additional arguments passed to `get_filter_logical()`.

Value

A filtered object, or a named list of filtered objects if `expanded = TRUE`.

Examples

```
library(S7)
df <- data.frame(
 category = rep(letters[1:3], each = 4),
 value = 1:12,
 date = as.Date('2024-01-01') + 0:11
)
filters <- list(
  category = c("a", "b"),
  value = c(3, 11)
)

# Apply filters with logical AND
apply_filters(df, filters, filter_combine_method = "and")

# Apply filters with logical OR
apply_filters(df, filters, filter_combine_method = "or")

# Get expanded filters
apply_filters(df, filters, expanded = TRUE)
```

`args_filter_input` *Derive Arguments for* **shiny** *Inputs*

Description

Provides the appropriate function arguments for the input function selected by `filterInput()` or `updateFilterInput()`.

Usage

```
args_filter_input(x, ...)

args_update_filter_input(x, ...)
```

Arguments

`x`	The object being passed to `filterInput()` or `updateFilterInput()`.
`...`	Additional arguments passed to the method. See details.

Details

The following arguments are supported in `...`:

range *(Date, POSIXt)*. Logical. If `TRUE`, `args_filter_input()` will provide the arguments for range date inputs. Only applies when `x` is of class `Date` or `POSIXt`.

textbox *(character)*. Logical. If `FALSE` (the default), `args_filter_input()` will provide the arguments for select inputs.

choices_asis *(character, factor, logical)*. Logical. If TRUE, the choices provided to select inputs will not be modified. If FALSE (the default), duplicate values will be removed and the choices will be sorted. Only applies when x is of class character, factor, or logical.

server If TRUE, indicates that the choices will be provided server-side. In this case, arguments are not computed for args_filter_input(). Ignored in args_update_filter_input().

args_unique An optional named list of arguments passed to unique(), called when x is a *character*, *factor*, or *logical*, textbox = FALSE, and choices_asis = FALSE.

args_sort An optional named list of arguments passed to sort(), which is called after unique().

Value

A named list of arguments for a **shiny** input function

Examples

```
args_filter_input(iris$Petal.Length)
```

---

call_input_function *Prepare and Evaluate Input Function and Arguments*

Description

Internal function used to prepare input arguments using args_filter_input(), and gracefully pass them to provided input function.

Usage

```
call_filter_input(x, .f, ...)

call_update_filter_input(x, .f, ...)
```

Arguments

x	The object being passed to filterInput().
.f	The input function to be called.
...	Arguments passed to either args_filter_input() or provided input function.

Details

call_filter_input() and call_update_filter_input() are used when customizing **shinyfilters**. For more, see vignette("customizing-shinyfilters").

Value

The result of calling the provided input function.

Examples

```
library(S7)
library(shiny)
# call_filter_input() is used inside filterInput() methods
method(filterInput, class_numeric) <- function(x, ...) {
  call_filter_input(x, sliderInput, ...)
}

# call_update_filter_input() is used inside updateFilterInput() methods
method(updateFilterInput, class_numeric) <- function(x, ...) {
  call_update_filter_input(x, updateSliderInput, ...)
}
```

`filterInput` *Create a* **shiny** *Input*

Description

Selects and creates a **shiny** input based the type of object `x` and other arguments.

Usage

```
filterInput(x, ...)
```

Arguments

`x`	The object used to create the input.
`...`	Arguments used for input selection or passed to the selected input. See details.

Details

The following arguments passed to `...` are supported:

area *(character)*. Logical. Controls whether to use textAreaInput (`TRUE`) or textInput (`FALSE`, default). Only applies when `textbox` is `TRUE`.

radio *(character, factor, list, logical)*. Logical. Controls whether to use radioButtons (`TRUE`) or a dropdown input (`FALSE`, default). For character vectors, `radio` only applies if `textbox` is `FALSE`, the default.

range *(Date, POSIXt)*. Logical. Controls whether to use dateRangeInput (`TRUE`) or dateInput (`FALSE`, default).

selectize *(character, factor, list, logical)*. Logical. Controls whether to use selectizeInput (`TRUE`) or selectInput (`FALSE`, default). For character vectors, `selectize` only applies if `textbox` is `FALSE`, the default.

slider *(numeric)*. Logical. Controls whether to use sliderInput (`TRUE`) or numericInput (`FALSE`, default).

textbox *(character)*. Logical. Controls whether to use a text input (TRUE) or a dropdown input (FALSE, default).

ns An optional namespace created by NS(). Useful when using filterInput() on a data.frame inside a **shiny** module.

Remaining arguments passed to ... are passed to the args_filter_input() or the selected input function.

Value

One of the following **shiny** inputs is returned, based on the type of object passed to x, and other specified arguments. See vignette("filter-input-catalog") for the full list of examples.

Value	x	Arguments
dateInput	Date, POSIXt	*default*
dateRangeInput	Date, POSIXt	range = TRUE
numericInput	numeric	*default*
radioButtons	character, factor, list, logical	radio = TRUE
selectInput	character, factor, list, logical	*default*
selectizeInput	character, factor, list, logical	selectize = TRUE
sliderInput	numeric	slider = TRUE
textAreaInput	character	textbox = TRUE, area = TRUE
textInput	character	textbox = TRUE

Examples

```
library(shiny)

ui <- fluidPage(
  sidebarLayout(
   sidebarPanel(
    # Create a filterInput() inside a shiny app:
    filterInput(
    x = letters,
    inputId = "letter",
    label = "Pick a letter:"
    )
   ),
   mainPanel(
    textOutput("selected_letter")
   )
  )
)

server <- function(input, output, session) {
  output$selected_letter <- renderText({
   paste("You selected:", input$letter)
  })
}
```

```
shinyApp(ui, server)
```

`get_filter_logical` *Compute a Filter Predicate*

Description

Computes a logical vector indicating which elements of `x` match the filter criteria specified by `val`.

Usage

```
get_filter_logical(x, val, ...)
```

Arguments

`x`	An object to filter; typically a data.frame.
`val`	The filter criteria.
`...`	Arguments passed to methods. See details.

Details

The following arguments are supported in `...`:

column When `x` is a data.frame, `column` is the name of the column intended to be filtered.

comparison When `x` is a numeric or Date and `val` is a length-**one** numeric or Date, `comparison` is the function used to compare `x` with `val`. The default is `<=`.

gte When `x` is a numeric or Date and `val` is a length-**two** numeric or Date, `gte` controls whether to use `>=` (`TRUE`, default) or `>` (`FALSE`) on `val[[1]]`.

lte When `x` is a numeric or Date and `val` is a length-**two** numeric or Date, `lte` controls whether to use `<=` (`TRUE`, default) or `<` (`FALSE`) on `val[[2]]`.

Value

A logical vector indicating which elements of `x` match the filter criteria specified by `val`.

Examples

```
df <- data.frame(
  category = rep(letters[1:3], each = 4),
  value = 1:12,
  date = Sys.Date() + 0:11
)

# Filter character column
get_filter_logical(df, c("a", "b"), column = "category")
```

```
# Filter numeric column with single value
get_filter_logical(df, 5, column = "value", comparison = `<=`)

# Filter numeric column with range
get_filter_logical(df, c(3, 8), column = "value", gte = TRUE, lte = FALSE)
```

`get_input_ids` *Retrieve the Ids of Input Objects*

Description

Returns the (unnamespaced) ids of the inputs for the provided object.

Usage

```
get_input_ids(x, ...)
```

Arguments

`x`	An object for which to retrieve input ids; typically a data.frame.
`...`	Passed onto methods.

Value

A character vector of input ids.

Examples

```
df <- data.frame(
  name = c("Alice", "Bob"),
  age = c(25, 30),
  completed = c(TRUE, FALSE)
)

get_input_ids(df)
```

get_input_labels *Retrieve the Labels of Input Objects*

Description

Returns the labels of the **shiny** inputs for the provided object.

Usage

```
get_input_labels(x, ...)
```

Arguments

x	An object for which to retrieve input labels; typically a data.frame.
...	Passed onto methods.

Value

A character vector of input labels

Examples

```
df <- data.frame(
  name = c("Alice", "Bob"),
  age = c(25, 30),
  completed = c(TRUE, FALSE)
)

get_input_labels(df)
```

get_input_values *Get Multiple Values from a* **shiny** *Input Object*

Description

Retrieves multiple input values from a **shiny** `input` object based on the names provided in `x`.

Usage

```
get_input_values(input, x, ...)
```

Arguments

input	A **shiny** `input` object, i.e., the `input` argument to the shiny server.
x	A character vector of input names, or a data.frame whose column names are converted to input names via `get_input_ids()`.
...	Passed onto methods.

Value

A named list of input values corresponding to the names in x.

Examples

```
library(shiny)
df <- data.frame(
  name = c("Alice", "Bob"),
  age = c(25, 30),
  completed = c(TRUE, FALSE)
)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      filterInput(df)
    ),
    mainPanel(
      verbatimTextOutput("output_all"),
      verbatimTextOutput("output_subset")
    )
  )
)
server <- function(input, output, session) {
  output$output_all <- renderPrint({
    get_input_values(input, df)
  })
  output$output_subset <- renderPrint({
    get_input_values(input, c("name", "completed"))
  })
}
shinyApp(ui, server)
```

---

serverFilterInput *Run the backend server for filterInput*

Description

Run the backend server for filterInput

Usage

```
serverFilterInput(
  x,
  input,
  filter_combine_method = "and",
  args_apply_filters = NULL,
  ...
)
```

Arguments

x	An object being filtered; typically a data.frame.
input	A **shiny** input object, or a reactive that resolves to a list of named values.
filter_combine_method	A string or function indicating how to combine multiple filters. If a string, it can be "and" (or "&") for logical AND, or "or" (or "|") for logical OR. If a function, it should take two logical vectors and return a combined logical vector.
args_apply_filters	A named list of additional arguments passed to apply_filters().
...	Additional arguments passed to updateFilterInput().

Value

A reactiveValues list with a single element, input_values, which contains the current filter input values as a named list.

Examples

```
library(bslib)
library(DT)
library(S7)
library(shiny)

must_use_radio <- new_S3_class(
  class = "must_use_radio",
  constructor = function(.data) .data
)
method(filterInput, must_use_radio) <- function(x, ...) {
  call_filter_input(x, radioButtons, ...)
}
method(updateFilterInput, must_use_radio) <- function(x, ...) {
  call_update_filter_input(x, updateRadioButtons, ...)
}

use_radio <- function(x) {
  structure(x, class = unique(c("must_use_radio", class(x))))
}

df_shared <- data.frame(
  x = letters,
  y = use_radio(sample(c("red", "green", "blue"), 26, replace = TRUE)),
  z = round(runif(26, 0, 3.5), 2),
  q = sample(Sys.Date() - 0:7, 26, replace = TRUE)
)

filters_ui <- function(id) {
  ns <- NS(id)
  filterInput(
   x = df_shared,
   range = TRUE,
```

```
   selectize = TRUE,
   slider = TRUE,
   multiple = TRUE,
   ns = ns
  )
}

filters_server <- function(id) {
  moduleServer(id, function(input, output, session) {
   # serverFilterInput() returns a observe() expressionc
   serverFilterInput(df_shared, input = input, range = TRUE)
  })
}

ui <- page_sidebar(
  sidebar = sidebar(filters_ui("demo")),
  DTOutput("df_full"),
  verbatimTextOutput("input_values"),
  DTOutput("df_filt")
)

server <- function(input, output, session) {
  res <- filters_server("demo")
  output$df_full <- renderDT(datatable(df_shared))
  output$input_values <- renderPrint(res$input_values)
  output$df_filt <- renderDT(datatable(apply_filters(
   df_shared,
   res$input_values
  )))
}

shinyApp(ui, server)
```

`updateFilterInput` *Update a* **shiny** *Input*

Description

Updates a **shiny** input based the type of object `x` and other arguments.

Usage

```
updateFilterInput(x, ...)
```

Arguments

`x`	The object used to create the input.
`...`	Arguments used for input selection or passed to the selected input update function. See details.

Details

The following arguments passed to ... are supported:

area *(character)*. Logical. Controls whether to use updateTextAreaInput (TRUE) or updateTextInput (FALSE, default). Only applies when textbox is TRUE.

radio *(character, factor, list, logical)*. Logical. Controls whether to use updateRadioButtons (TRUE) or a dropdown input update function (FALSE, default). For character vectors, radio only applies if textbox is FALSE, the default.

range *(Date, POSIXt)*. Logical. Controls whether to use updateDateRangeInput (TRUE) or updateDateInput (FALSE, default).

selectize *(character, factor, list, logical)*. Logical. Controls whether to use updateSelectizeInput (TRUE) or updateSelectInput (FALSE, default). For character vectors, selectize only applies if textbox is FALSE, the default.

slider *(numeric)*. Logical. Controls whether to use updateSliderInput (TRUE) or updateNumericInput (FALSE, default).

textbox *(character)*. Logical. Controls whether to update a text input (TRUE) or a dropdown input (FALSE, default).

Remaining arguments passed to ... are passed to `args_update_filter_input()` or the selected input update function.

Value

The result of the following **shiny** input updates is returned, based on the type of object passed to x, and other specified arguments.

Value	x	Arguments
updateDateInput	Date, POSIXt	*default*
updateDateRangeInput	Date, POSIXt	range = TRUE
updateNumericInput	numeric	*default*
updateRadioButtons	character, factor, list, logical	radio = TRUE
updateSelectInput	character, factor, list, logical	*default*
updateSelectizeInput	character, factor, list, logical	selectize = TRUE
updateSliderInput	numeric	slider = TRUE
updateTextAreaInput	character	textbox = TRUE, area = TRUE
updateTextInput	character	textbox = TRUE

Examples

```
library(shiny)

fruits <- list(
 "a" = c("apples", "avocados"),
 "b" = c("bananas", "blueberries"),
 "c" = c("cherries", "cantaloupe")
)
```

```
ui <- fluidPage(
 sidebarLayout(
  sidebarPanel(
   filterInput(
    x = letters[1:3],
    inputId = "letter",
    label = "Pick a letter:",
       multiple = TRUE
   ),
   filterInput(
    x = fruits,
    inputId = "fruits",
    label = "Pick a fruit:"
   )
  ),
   mainPanel()
 )
)

server <- function(input, output, session) {
 observe({
  fruits_filtered <- fruits
  if (!is.null(input$letter) && length(input$letter) != 0L) {
   fruits_filtered <- fruits[input$letter]
  }
  # Call updateFilterInput() inside the shiny server:
  updateFilterInput(x = fruits_filtered, inputId = "fruits")
 })
}
shinyApp(ui, server)
```

Index

apply_filters, 2
apply_filters(), *11*
args_filter_input, 3
args_filter_input(), *4*, *6*
args_update_filter_input
        *(*args_filter_input*)*, 3
args_update_filter_input(), *13*

call_filter_input
        *(*call_input_function*)*, 4
call_input_function, 4
call_update_filter_input
        *(*call_input_function*)*, 4

dateInput, *5*, *6*
dateRangeInput, *5*, *6*

filterInput, 5
filterInput(), *3*, *4*

get_filter_logical, 7
get_filter_logical(), *2*
get_input_ids, 8
get_input_ids(), *9*
get_input_labels, 9
get_input_values, 9

NS(), *6*
numericInput, *5*, *6*

radioButtons, *5*, *6*

selectInput, *5*, *6*
selectizeInput, *5*, *6*
serverFilterInput, 10
sliderInput, *5*, *6*
sort(), *4*

textAreaInput, *5*, *6*
textInput, *5*, *6*

unique(), *4*

updateDateInput, *13*
updateDateRangeInput, *13*
updateFilterInput, 12
updateFilterInput(), *3*, *11*
updateNumericInput, *13*
updateRadioButtons, *13*
updateSelectInput, *13*
updateSelectizeInput, *13*
updateSliderInput, *13*
updateTextAreaInput, *13*
updateTextInput, *13*