Package 'teal.slice'

February 6, 2024

Type Package

Title Filter Module for 'teal' Applications

Version 0.5.0

Date 2024-01-31

Description Data filtering module for 'teal' applications. Allows for interactive filtering of data stored in 'data.frame' and 'MultiAssayExperiment' objects. Also displays filtered and unfiltered observation counts.

License Apache License 2.0

URL https://insightsengineering.github.io/teal.slice/, https://github.com/insightsengineering/teal.slice/

BugReports https://github.com/insightsengineering/teal.slice/issues

Depends R (>= 4.0)

Imports bslib (>= 0.4.0), checkmate (>= 2.1.0), dplyr (>= 1.0.5), grDevices, htmltools (>= 0.5.4), jsonlite, lifecycle (>= 0.2.0), logger (>= 0.2.0), methods, plotly (>= 4.9.2.2), R6 (>= 2.2.0), shiny (>= 1.6.0), shinycssloaders (>= 1.0.0), shinyjs, shinyWidgets (>= 0.6.2), teal.data (>= 0.4.0), teal.logger (>= 0.1.1), teal.widgets (>= 0.4.0), utils

Suggests knitr (>= 1.42), MultiAssayExperiment, SummarizedExperiment, testthat (>= 3.1.5), withr (>= 2.1.0)

VignetteBuilder knitr

RdMacros lifecycle

Config/Needs/verdepcheck rstudio/shiny, rstudio/bslib, mllg/checkmate, tidyverse/dplyr, rstudio/htmltools, jeroen/jsonlite, r-lib/lifecycle, daroczig/logger, plotly/plotly, r-lib/R6, daattali/shinycssloaders, daattali/shinyjs, dreamRs/shinyWidgets, insightsengineering/teal.data, insightsengineering/teal.logger, insightsengineering/teal.widgets, yihui/knitr, bioc::MultiAssayExperiment, bioc::SummarizedExperiment, r-lib/testthat, r-lib/withr

Config/Needs/website insightsengineering/nesttemplate

Encoding UTF-8

Language en-US

RoxygenNote 7.3.1

NeedsCompilation no

Author Dawid Kaledkowski [aut, cre] (<https://orcid.org/0000-0001-9533-457X>),
Pawel Rucki [aut],
Aleksander Chlebowski [aut] (<https://orcid.org/0000-0001-5018-6294>),
Andre Verissimo [aut] (<https://orcid.org/0000-0002-2212-339X>),
Kartikeya Kirar [aut],
Marcin Kosinski [aut],
Chendi Liao [rev],
Dony Unardi [rev],
Andrew Bates [aut],
Mahmoud Hallal [aut],
Nikolas Burkoff [aut],
Maciej Nasinski [aut],
Konrad Pagacz [aut],
Junlue Zhao [aut],
F. Hoffmann-La Roche AG [cph, fnd]

Maintainer Dawid Kaledkowski <dawid.kaledkowski@roche.com>

Repository CRAN

Date/Publication 2024-02-06 17:40:02 UTC

R topics documented:

`FilterPanelAPI` *Class to encapsulate the API of the filter panel of a teal app*

Description

An API class for managing filter states in a teal application's filter panel.

Details

The purpose of this class is to encapsulate the API of the filter panel in a new class `FilterPanelAPI` so that it can be passed and used in the server call of any module instead of passing the whole `FilteredData` object.

This class is supported by methods to set, get, remove filter states in the filter panel API.

Methods

Public methods:

Method `new()`**:** Initialize a `FilterPanelAPI` object.

Usage:

```
FilterPanelAPI$new(datasets)
```

Arguments:

`datasets` (`FilteredData`)

Method `get_filter_state()`**:** Gets the reactive values from the active `FilterState` objects of the `FilteredData` object.
Gets all active filters in the form of a nested list. The output list is a compatible input to `set_filter_state`.

Usage:

```
FilterPanelAPI$get_filter_state()
```

Returns: `list` with named elements corresponding to `FilteredDataset` objects with active filters.

Method `set_filter_state()`**:** Sets active filter states.

Usage:

```
FilterPanelAPI$set_filter_state(filter)
```

Arguments:

`filter` (`teal_slices`)

Returns: `NULL`, invisibly.

Method `remove_filter_state()`**:** Remove one or more `FilterState` of a `FilteredDataset` in the `FilteredData` object.

Usage:

```
FilterPanelAPI$remove_filter_state(filter)
```

Arguments:

`filter` (`teal_slices`) specifying `FilterState` objects to remove; `teal_slices` may contain only `dataname` and `varname`, other elements are ignored

Returns: `NULL`, invisibly.

Method `clear_filter_states()`**:** Remove all `FilterStates` of the `FilteredData` object.

Usage:

```
FilterPanelAPI$clear_filter_states(datanames)
```

Arguments:

`datanames` (`character`) `datanames` to remove their `FilterStates`; omit to remove all `FilterStates` in the `FilteredData` object

Returns: `NULL`, invisibly.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FilterPanelAPI$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(shiny)

fd <- init_filtered_data(list(iris = iris))
fpa <- FilterPanelAPI$new(fd)

# get the actual filter state --> empty named list
isolate(fpa$get_filter_state())

# set a filter state
set_filter_state(
  fpa,
  teal_slices(
   teal_slice(dataname = "iris", varname = "Species", selected = "setosa", keep_na = TRUE)
  )
)

# get the actual filter state --> named list with filters
isolate(fpa$get_filter_state())

# remove all_filter_states
fpa$clear_filter_states()

# get the actual filter state --> empty named list
isolate(fpa$get_filter_state())
```

`filter_state_api` *Managing* `FilteredData` *states*

Description

[Experimental]

Set, get and remove filter states of `FilteredData` object.

Usage

```
set_filter_state(datasets, filter)

get_filter_state(datasets)

remove_filter_state(datasets, filter)

clear_filter_states(datasets, force = FALSE)
```

Arguments

datasets	(FilteredData) object to store filter state and filtered datasets, shared across modules see FilteredData for details
filter	(teal_slices) specify filters in place on app start-up
force	(logical(1)) flag specifying whether to include anchored filter states.

Value

- set_*, remove_* and clear_filter_state return NULL invisibly
- get_filter_state returns a named teal_slices object containing a teal_slice for every existing FilterState

See Also

teal_slice

Examples

```
datasets <- init_filtered_data(list(iris = iris, mtcars = mtcars))
fs <- teal_slices(
 teal_slice(dataname = "iris", varname = "Species", selected = c("setosa", "versicolor")),
  teal_slice(dataname = "iris", varname = "Sepal.Length", selected = c(5.1, 6.4)),
  teal_slice(dataname = "mtcars", varname = "gear", selected = c(4, 5)),
  teal_slice(dataname = "mtcars", varname = "carb", selected = c(4, 10))
)

# set initial filter state
set_filter_state(datasets, filter = fs)

# get filter state
get_filter_state(datasets)

# modify filter state
set_filter_state(
  datasets,
  teal_slices(
   teal_slice(dataname = "iris", varname = "Species", selected = "setosa", keep_na = TRUE)
  )
)
```

```
# remove specific filters
remove_filter_state(
  datasets,
  teal_slices(
    teal_slice(dataname = "iris", varname = "Species"),
    teal_slice(dataname = "mtcars", varname = "gear"),
    teal_slice(dataname = "mtcars", varname = "carb")
  )
)

# remove all states
clear_filter_states(datasets)


if (requireNamespace("MultiAssayExperiment", quietly = TRUE)) {
  # Requires MultiAssayExperiment from Bioconductor
  data(miniACC, package = "MultiAssayExperiment")

  datasets <- init_filtered_data(list(mae = miniACC))
  fs <- teal_slices(
    teal_slice(
      dataname = "mae", varname = "years_to_birth", selected = c(30, 50),
      keep_na = TRUE, keep_inf = FALSE
    ),
    teal_slice(
      dataname = "mae", varname = "vital_status", selected = "1",
      keep_na = FALSE
    ),
    teal_slice(
      dataname = "mae", varname = "gender", selected = "female",
      keep_na = TRUE
    ),
    teal_slice(
      dataname = "mae", varname = "ARRAY_TYPE", selected = "",
      keep_na = TRUE, experiment = "RPPAArray", arg = "subset"
    )
  )

  # set initial filter state
  set_filter_state(datasets, filter = fs)

  # get filter state
  get_filter_state(datasets)

  # modify filter state
  set_filter_state(
    datasets,
    teal_slices(
      teal_slice(dataname = "mae", varname = "years_to_birth", selected = c(40, 60))
    )
  )
```

```
  # remove specific filters
  remove_filter_state(
    datasets,
    teal_slices(
      teal_slice(dataname = "mae", varname = "years_to_birth"),
      teal_slice(dataname = "mae", varname = "vital_status")
    )
  )

  # remove all states
  clear_filter_states(datasets)
}
```

`get_filter_expr` *Gets filter expression for multiple* `datanames` *taking into account its order.*

Description

[Stable]

To be used in `Show R Code` button.

Usage

```
get_filter_expr(datasets, datanames = datasets$datanames())
```

Arguments

`datasets`	(`FilteredData`)
`datanames`	(`character`) vector of dataset names

Value

A character string containing all subset expressions.

`init_filtered_data` *Initialize* `FilteredData`

Description

Function creates a `FilteredData` object.

Usage

```
init_filtered_data(x, join_keys = teal.data::join_keys(), code, check)
```

Arguments

x	(named list) of datasets.
join_keys	(join_keys) see teal.data::join_keys().
code	**[Deprecated]**
check	**[Deprecated]**

Value

Object of class FilteredData.

Examples

```
datasets <- init_filtered_data(list(iris = iris, mtcars = mtcars))
datasets
```

Index