Package ‘IDEAFilter’

October 12, 2022

**Type** Package

**Version** 0.1.2

**Title** Agnostic, Idiomatic Data Filter Module for Shiny

**Description** When added to an existing shiny app, users may subset any developer-chosen R data.frame on the fly. That is, users are empowered to slice & dice data by applying multiple (order specific) filters using the AND (&) operator between each, and getting real-time updates on the number of rows effected/available along the way. Thus, any downstream processes that leverage this data source (like tables, plots, or statistical procedures) will re-render after new filters are applied. The shiny module’s user interface has a 'minimalist' aesthetic so that the focus can be on the data & other visuals. In addition to returning a reactive (filtered) data.frame, 'IDEAFilter' as also returns 'dplyr' filter statements used to actually slice the data.

**License** AGPL-3

**URL** https://biogen-inc.github.io/IDEAFilter/

**BugReports** https://github.com/Biogen-Inc/IDEAFilter/issues

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**Imports** shiny, ggplot2, pillar (>= 1.5.0), crayon, RColorBrewer, shinyTime, purrr

**Suggests** shinytest, shinytest2, testthat, knitr, rmarkdown, spelling, dplyr

**Language** en-US

**NeedsCompilation** no

**Author** Aaron Clark [aut, cre] (<https://orcid.org/0000-0002-0123-0970>), Doug Kelkhoff [aut], Maya Gans [ctb], Jeff Thompson [ctb], Biogen [cph]

**Maintainer** Aaron Clark <clark.aaronchris@gmail.com>
Description

Shiny data filter module server function

Usage

shiny_data_filter(input, output, session, data, verbose = FALSE)

Arguments

- **input**: requisite shiny module field specifying incoming ui input reactiveValues
- **output**: requisite shiny module field capturing output for the shiny data filter ui
- **session**: requisite shiny module field containing the active shiny session
- **data**: a data.frame or reactive expression returning a data.frame to use as the input to the filter module
- **verbose**: a logical value indicating whether or not to print log statements out to the console

Value

A reactive expression which returns the filtered data wrapped in an additional class, "shiny-DataFilter_df". This structure also contains a "code" field which represents the code needed to generate the filtered data.

See Also

shiny_data_filter_ui
**Examples**

```r
if(all(c(interactive(), require("dplyr"), require("IDEAFilter")))) {
library(shiny)
library(IDEAFilter)
library(dplyr) # for data pre-processing and example data

# prep a new data.frame with more diverse data types
starwars2 <- starwars %>%
  mutate_if(~is.numeric(.) & all(Filter(Negate(is.na), .) %% 1 == 0), as.integer) %>%
  mutate_if(~is.character(.) & length(unique(.)) <= 25, as.factor) %>%
  mutate(is_droid = species == "Droid") %>%
  select(name, gender, height, mass, hair_color, eye_color, vehicles, is_droid)

# create some labels to showcase column select input
attr(starwars2$name, "label") <- "name of character"
attr(starwars2$gender, "label") <- "gender of character"
attr(starwars2$height, "label") <- "height of character in centimeters"
attr(starwars2$mass, "label") <- "mass of character in kilograms"
attr(starwars2$is_droid, "label") <- "whether character is a droid"

ui <- fluidPage(
  titlePanel("Filter Data Example"),
  fluidRow(
    column(8,
     verbatimTextOutput("data_summary"),
      verbatimTextOutput("data_filter_code")),
    column(4, shiny_data_filter_ui("data_filter")))
)

server <- function(input, output, session) {
  filtered_data <- callModule(
    shiny_data_filter,
    "data_filter",
    data = starwars2,
    verbose = FALSE)

  output$data_filter_code <- renderPrint(
    cat(gsub("%>%", "%>% \n ",
      gsub("\s{2,}", " ",
        paste0(
          capture.output(attr(filtered_data(), "code")),
          collapse = " ")))
  )

  output$data_summary <- renderPrint(
    if (nrow(filtered_data())) show(filtered_data())
    else "No data available"
  )
}

shinyApp(ui = ui, server = server)
}````
Index

shiny_data_filter, 2
shiny_data_filter_ui, 2