Package 'shinypivottabler'

January 6, 2023

Title Shiny Module to Create Pivot Tables

Version 1.2

Description Shiny Module to create, visualize, customize and export Excel-like pivot table.

License GPL-3

Encoding UTF-8

Depends R (>= 3.4)

RoxygenNote 7.2.1

Imports pivottabler (>= 1.5.0), shiny, openxlsx, colourpicker, htmltools

NeedsCompilation no

Author Benoit Thieurmel [aut, cre], Thibaut Dubois [aut]

Maintainer Benoit Thieurmel <bthieurmel@gmail.com>

Repository CRAN

Date/Publication 2023-01-06 10:30:02 UTC

R topics documented:

`shinypivottabler` *Shiny module to render and export pivot tables.*

Description

Shiny module to render and export pivot tables.

Usage

```
shinypivottabler(
  input,
  output,
  session,
  data,
  pivot_cols = NULL,
  indicator_cols = NULL,
  max_n_pivot_cols = 100,
  additional_expr_num = list(),
  additional_expr_char = list(),
  additional_combine = list(),
  theme = NULL,
  export_styles = TRUE,
  show_title = TRUE,
  initialization = NULL
)

shinypivottablerUI(id, app_colors = c("#59bb28", "#217346"), app_linewidth = 8)
```

Arguments

input	shiny input
output	shiny input
session	shiny input
data	data.frame / data.table. Initial data table.
pivot_cols	character (NULL). Columns to be used as pivot in rows and cols.
indicator_cols	character (NULL). Columns on which indicators will be calculated.
max_n_pivot_cols	numeric (100). Maximum unique values for a pivot_cols if pivot_cols = NULL
additional_expr_num	named list (list()). Additional computations to be allowed for quantitative vars.
additional_expr_char	named list (list()). Additional computations to be allowed for qualitative vars.
additional_combine	named list (list()). Additional combinations to be allowed.
theme	list (NULL). Theme to customize the output of the pivot table. Use HEX color rather than rgb for export style
export_styles	boolean (TRUE). Whether or not to apply styles (like the theme) when exporting to Excel.
show_title	boolean (TRUE). Whether or not to display the app title. Some styles may not be supported by Excel.
initialization	named list (NULL). Initialization parameters to display a table when launching the module. Available fields are :

- `rows`: Selected pivot rows.
- `cols`: Selected pivot columns.
- `target`, `combine target`: Selected target and combine_target columns..
- `idc`, `combine_idc`: Selected idc and combine_idc columns.
- `combine`: Selected combine operator.
- `format_digit`, `format_prefix`, `format_suffix`, `format_sep_thousands`, `format_decimal`: Selected formats for the table idc.
- `idcs`: idcs to be displayed (list of named list), see the example to get the fields.

`id`	`character`. An ID string
`app_colors`	`character`. Vector of two colors c("#59bb28", "#217346") (borders)
`app_linewidth`	`numeric`. Borders width

Value

Nothing. Just Start a Shiny module.

Examples

```
if (interactive()) {
require(shinypivottabler)
require(shiny)

# demo app
runApp(system.file("demo_app", package = "shinypivottabler"))

# create artificial dataset
n <- 1000000
data <- data.frame("gr1" = sample(c("A", "B", "C", "D"), size = n,
                                  prob = rep(1, 4), replace = T),
                   "gr2" = sample(c("E", "F", "G", "H"), size = n,
                                  prob = rep(1, 4), replace = T),
                   "gr3" = sample(c("I", "J", "K", "L"), size = n,
                                  prob = rep(1, 4), replace = T),
                   "gr4" = sample(c("M", "N", "O", "P"), size = n,
                                  prob = rep(1, 4), replace = T),
                   "value1" = 1:n,
                   "value2" = n:1)

# Minimal example

ui = shiny::fluidPage(
  shinypivottablerUI(id = "id")
)

server = function(input, output, session) {
  shiny::callModule(module = shinypivottabler,
                    id = "id",
```

```
                    data = data)
}

shiny::shinyApp(ui = ui, server = server)



# Complete example

initialization <- list(
  "rows" = "gr1",
  "cols" = "gr2",
  "target" = "gr3",
  "combine_target" = "gr4",
  "idc" = "Count",
  "combine_idc" = "Count",
  "combine" = "/",
  "idcs" = c(
      list(
        c("label" = "Init_variable_1",
          "target" = "gr3", "idc" = "Count",
          "nb_decimals" = 0,
          "sep_thousands" = " ",
          "sep_decimal" = ".",
          "prefix" = "",
          "suffix" = "",
          "combine" = "/",
          "combine_target" = "gr4",
          "combine_idc" = "Count")
       ),
       list(
         c("label" = "Init_variable_2",
           "target" = "gr3", "idc" = "Count")
       )
     )
)

theme <- list(
  fontName="Courier New, Courier",
  fontSize="1em",
  headerBackgroundColor = "red",
  headerColor = "#FFFFFF",
  cellBackgroundColor = "#FFFFFF",
  cellColor = "#000000",
  outlineCellBackgroundColor = "#C0C0C0",
  outlineCellColor = "#000000",
  totalBackgroundColor = "#59bb28",
  totalColor = "#000000",
  borderColor = "#404040"
)

ui = shiny::fluidPage(
  shinypivottablerUI(id = "id")
```

```
)

# we add two functions, one for quantitative variables (Q5) and
# one for qualitatives variables (the mode, with a custom function), and
# one possible combination (the modulo).
my_mode <- function(x) names(which.max(table(x)))

server = function(input, output, session) {
  shiny::callModule(module = shinypivottabler,
                    id = "id",
                    data = data,
                    pivot_cols = c("gr1", "gr2", "gr3", "gr4"),
                    additional_expr_num = list(
                  "Add_Q5" = "paste0('quantile(', target, ', probs = 0.05, na.rm = TRUE)')"
                    ),
                    additional_expr_char = list(
                      "Add_mode" = "paste0('my_mode(', target, ')')"
                    ),
                    additional_combine = c("Add_modulo" = "%%"),
                    theme = theme,
                    initialization = initialization)
}

shiny::shinyApp(ui = ui, server = server)

}
```

Index