Package 'vvshiny'

July 19, 2023

Title Create Complex Shiny Apps More Easily

Version 0.1.1

Description
Helper and Wrapper functions for making shiny dashboards more easily. Functions are made modular and lower level functions are exported as well, so many use-cases are supported.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports stats, dplyr, magrittr, purrr, rlang, stringr, forcats, ggplot2, ggpubr, scales, ggalluvial, shiny, shinycssloaders, shinydashboard, shinydashboardPlus, shinyWidgets, htmlwidgets, htmltools, plotly, DT

Suggests RColorBrewer, waiter, spsComps, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Corneel Den Hartogh [aut, cre] (<https://orcid.org/0000-0001-8347-7944>), Tomer Iwan [ctb], VU Analytics [cph]

Maintainer Corneel Den Hartogh <c.f.den.hartogh@vu.nl>

Repository CRAN

Date/Publication 2023-07-19 15:30:02 UTC

R topics documented:

`basic_plot` *Create a basic plot using ggplot*

Description

This function creates a basic plot with the help of ggplot.

Usage

```
basic_plot(
  df,
  x,
  y,
  color,
  xlab_setting,
  ylab_setting,
  ggplot_settings = ggplot_basic_settings(),
  legend_position = "none",
  scale_y = NULL
)
```

Arguments

df	A data frame containing the data to be plotted.
x	A string specifying the column name to be used as the x-axis variable.
y	A string specifying the column name to be used as the y-axis variable.
color	A string specifying the column name to be used as the fill variable.
xlab_setting	ggplot labels settings for x axes.
ylab_setting	ggplot labels settings for y axes.
ggplot_settings	Additional settings for the ggplot.
legend_position	A string specifying the position of the legend.
scale_y	Optional ggplot2 scale function to modify the y axis.

Value

A ggplot plot.

Examples

```
df <- data.frame(x_var = rnorm(100),
                 y_var = rnorm(100),
                 color_var = sample(c("Red", "Blue"),
                 100,
                 replace = TRUE))
xlab_setting <- ggplot2::xlab("x label")
ylab_setting <- ggplot2::ylab("y label")
ggplot_instellingen <- ggplot2::geom_point()
scale_y <- ggplot2::scale_y_continuous()
basic_plot(df, "x_var", "y_var", "color_var", xlab_setting,
           ylab_setting, ggplot_instellingen, "none", scale_y)
```

bind_both *Bind both*

Description

This function binds two dataframes row-wise and performs additional manipulations depending on the 'type'. The function also reorders the factor levels of the specified facet variable.

Usage

```
bind_both(
  dfLeft,
  dfRight,
  id = "bench",
  y_left = NULL,
  y_right = NULL,
  facet_var = rlang::sym("VIS_Groep"),
  facet_name_var = rlang::sym("VIS_Groep_naam")
)
```

Arguments

dfLeft	A dataframe to be combined
dfRight	A dataframe to be combined
id	An identifier string specifying the type of operation
y_left	A character vector specifying the column to be used for the left dataframe
y_right	A character vector specifying the column to be used for the right dataframe
facet_var	A symbol specifying the variable to be used for faceting
facet_name_var	A symbol specifying the variable to be used for the facet name

Value

A dataframe obtained by binding dfLeft and dfRight, with additional transformations applied

Examples

```
df1 <- data.frame(x = 1:5, y = rnorm(5), VIS_Groep_naam = "One")
df2 <- data.frame(x = 6:10, y = rnorm(5), VIS_Groep_naam = "Two")
df_both <- bind_both(df1, df2, id = "test",
                     y_left = "y", y_right = "y",
                     facet_var = rlang::sym("x"))
```

`bind_both_table` *Bind both table*

Description

This function joins two summarized dataframes and relocates y_left before y_right. The function also sets the VIS_Groep value to 'left' for the right dataframe.

Usage

```
bind_both_table(dfLeft_summ, dfRight_summ, y_left, y_right)
```

Arguments

`dfLeft_summ`	A summarized dataframe to be joined
`dfRight_summ`	A summarized dataframe to be joined
`y_left`	A character vector specifying the column to be relocated before y_right
`y_right`	A character vector specifying the column after which y_left will be relocated

Value

A dataframe obtained by joining dfLeft_summ and dfRight_summ, with y_left relocated before y_right

Examples

```
df1 <- data.frame(
  VIS_Groep = "a",
  x = c("a", "b"),
  y1 = 1:2
)
df2 <- data.frame(
  VIS_Groep = "b",
  x = c("a", "b"),
  y2 = 3:4
)

df_both <- bind_both_table(df1, df2, "y1", "y2")
```

clean_pltly_legend *Clean the legend of a plotly object*

Description

This function cleans the legend of a plotly object by removing unnecessary duplication. It is specifically designed to work around a bug that causes facet_wrap to create a separate legend entry for each facet.

Usage

```
clean_pltly_legend(pltly_obj, new_legend = c())
```

Arguments

pltly_obj	A plotly object with a legend to be cleaned.
new_legend	An optional vector of strings specifying new legend entries. Default is an empty vector.

Value

The input plotly object with its legend cleaned.

display_name *Get a user-friendly display name*

Description

This function provides a user-friendly name for a column based on a mapping table, if available.

Usage

```
display_name(col_name, mapping_table)
```

Arguments

col_name	A string specifying the name of the column.
mapping_table	A named list with as name the original colum name and as value the display name

Value

A string containing the user-friendly name for the column.

Examples

```
  mapping <- list(
col1 = "Column 1",
col2 = "Column 2"
)
display_name("col1", mapping)
```

`dropdownTabDirect` *dropdownTabDirect function*

Description

Dropdown that is actually a link to a tab.

Usage

```
dropdownTabDirect(
  type = c("messages", "notifications", "tasks"),
  tab_name,
  title,
  icon = NULL,
  .list = NULL,
  header = NULL
)
```

Arguments

`type`	A character vector of either "messages", "notifications", "tasks". Default is c("messages", "notifications", "tasks").
`tab_name`	The name of the tab to link to.
`title`	The title of the dropdown.
`icon`	The icon to use in the dropdown. If NULL, defaults will be set based on type.
`.list`	A list of items to add to the dropdown.
`header`	The header for the dropdown.

Value

A dropdown menu in the form of an HTML list, where clicking the dropdown directs to a specific tab.

Examples

```
dropdownTabDirect(type = "messages", tab_name = "Tab1", title = "Interesting tab")
```

dropdownTabMenu *dropdownTabMenu function*

Description

Dropdown that is actually more of a menu with adapted tasks.

Usage

```
dropdownTabMenu(
  ...,
  type = c("messages", "notifications", "tasks"),
  title = NULL,
  icon = NULL,
  .list = NULL,
  header = NULL
)
```

Arguments

...	additional arguments.
type	A character vector of either "messages", "notifications", "tasks". Default is c("messages", "notifications", "tasks").
title	The title of the dropdown.
icon	The icon to use in the dropdown. If NULL, defaults will be set based on type.
.list	A list of items to add to the dropdown.
header	The header for the dropdown.

Value

A dropdown menu in the form of an HTML list.

Examples

```
dropdownTabMenu(type = "messages", title = "Category tab items")
```

filter_with_lists *Filter with lists*

Description

This function filters a dataframe using a list with column and one or more values.

Usage

```
filter_with_lists(df, filters)
```

Arguments

df	A dataframe to be filtered
filters	A list of lists containing column names in the first element and a list their corresponding values for filtering in the second element

Value

A dataframe filtered based on the input filters

Examples

```
df <- dplyr::tibble(
  VIS_Groep = sample(c("Group1", "Group2", "Group3"), 100, replace = TRUE),
  VIS_Groep_naam = sample(c("Name1", "Name2", "Name3"), 100, replace = TRUE),
  var1 = sample(c("A", "B", "C"), 100, replace = TRUE),
  var2 = rnorm(100),
  color_var = sample(c("Red", "Blue", "Green"), 100, replace = TRUE)
)
filters = list(c("var1", c("A", "B")))
dfFiltered <- filter_with_lists(df, filters)
```

gantt_app *Gantt Chart Shiny App*

Description

Gantt Chart Shiny App

Usage

```
gantt_app(df, df_config_gantt, id = "gantt")
```

Arguments

df	Data frame for Gantt chart
df_config_gantt	Config data frame for Gantt chart
id	Module ID for Gantt chart

Value

Shiny app object

Examples

```
df <- dplyr::tribble( ~OPL_Onderdeel_CROHO_examen, ~OPL_Onderdeel_CROHO_instroom,
~OPL_CBS_Label_rubriek_examen, ~OPL_CBS_Label_rubriek_instroom, "GEDRAG EN MAATSCHAPPIJ",
"GEZONDHEIDSZORG", "sociale geografie", "(huis)arts, specialist, geneeskunde",
"GEDRAG EN MAATSCHAPPIJ", "GEDRAG EN MAATSCHAPPIJ", "sociale geografie", "sociale geografie",
"GEDRAG EN MAATSCHAPPIJ", "RECHT", "sociale geografie", "notariaat", "RECHT", "RECHT",
"notariaat", "notariaat", "TAAL EN CULTUUR", "RECHT", "niet westerse talen en culturen",
"notariaat")

df_config_gantt <- dplyr::tribble( ~Categorie, ~Veldnaam, ~Veldnaam_gebruiker, ~input_var,
~target_var, ~title_start, ~title_end, ~position_y_label, "Doorstroom vanuit B ",
"OPL_Onderdeel_CROHO_examen","B Croho sector", "OPL_Onderdeel_CROHO_examen",
"OPL_Onderdeel_CROHO_instroom","Waar stromen", "Bachelor gediplomeerden naar toe?","right",
"Doorstroom vanuit B ","OPL_CBS_Label_rubriek_examen","B ISCED-F Rubriek",
"OPL_CBS_Label_rubriek_examen","OPL_CBS_Label_rubriek_instroom","Waar stromen",
"Bachelor gediplomeerden naar toe?","right", "Instroom bij M", "OPL_Onderdeel_CROHO_instroom",
"M Croho sector", "OPL_Onderdeel_CROHO_instroom", "OPL_Onderdeel_CROHO_examen",
"Waarvandaan stromen ", " Master studenten in?", "left", "Instroom bij M",
"OPL_CBS_Label_rubriek_instroom", "M ISCED-F Rubriek", "OPL_CBS_Label_rubriek_instroom",
"OPL_CBS_Label_rubriek_examen", "Waarvandaan stromen ", " Master studenten in?", "left" )
```

---

gantt_plot *Create a Gantt plot using ggplot and plotly*

Description

This function creates a Gantt plot with the help of ggplot and plotly.

Usage

```
gantt_plot(df, x, xend, split_var, title, position_label_y)
```

Arguments

df	A data frame containing the data to be plotted.
x	A string specifying the column name to be used as the x-axis variable.
xend	A string specifying the column name to be used as the end of the x-axis variable.
split_var	A string specifying the column name to be used as the splitting variable.
title	A string specifying the title of the plot.
position_label_y	A string specifying the position of y-axis labels.

Value

A Gantt plot.

---

ggplotly_with_legend *Make ggplotly and add legend with color as title*

Description

This function creates a Plotly version of a ggplot2 object and adds a legend with the user-friendly name of the color variable as its title.

Usage

```
ggplotly_with_legend(plot, color, mapping_table)
```

Arguments

plot	A ggplot object.
color	A string specifying the column name to be used as the color variable.
mapping_table	A named list with as name the original colum name and as value the display name

Value

A plotly object with a formatted legend.

Examples

```
df <- data.frame(x_var = rnorm(100),
                 y_var = rnorm(100),
                 color_var = sample(c("Red", "Blue"),
                 100,
                 replace = TRUE))
xlab_setting <- ggplot2::xlab("x label")
ylab_setting <- ggplot2::ylab("y label")
ggplot_instellingen <- ggplot2::geom_point()
```

```
scale_y <- ggplot2::scale_y_continuous()
plot <- basic_plot(df, "x_var", "y_var", "color_var", xlab_setting,
                   ylab_setting, ggplot_instellingen, "none", scale_y)
mapping_table <- list(color_var = "user friendly name var")
plotly_object <- ggplotly_with_legend(plot, "color_var", mapping_table)
```

---

ggplot_basic_settings *Set ggplot basic settings*

Description

Basic ggplot settings are put in a list and returned

Usage

```
ggplot_basic_settings()
```

Value

A list with ggplot settings

---

grid_boxplots *Grid boxplot*

Description

Function for creating grid boxplots for either benchmark or comparison across variables.

Usage

```
grid_boxplots(
  df,
  x,
  color,
  y,
  id,
  y_left = NULL,
  y_right = NULL,
  facet_var = rlang::sym("VIS_Groep"),
  facet_name_var = rlang::sym("VIS_Groep_naam")
)
```

Arguments

df	The data frame used to create the plot.
x	The variable used on the x-axis of the plot.
color	The variable used to color the points or bars in the plot.
y	The variable used on the y-axis of the plot.
id	The identifier for selecting the data frame source.
y_left	The variable used on the left y-axis when creating a comparative plot.
y_right	The variable used on the right y-axis when creating a comparative plot.
facet_var	The variable used for facet wrapping.
facet_name_var	The name of the variable used for facet wrapping.

Value

A ggplot object.

---

grid_histograms *Grid histogram*

Description

Function for creating grid histograms.

Usage

```
grid_histograms(
  df,
  x,
  color,
  y,
  id,
  y_left = NULL,
  y_right = NULL,
  facet_var = rlang::sym("VIS_Groep"),
  facet_name_var = rlang::sym("VIS_Groep_naam")
)
```

Arguments

df	The data frame used to create the plot.
x	The variable used on the x-axis of the plot.
color	The variable used to color the points or bars in the plot.
y	The variable used on the y-axis of the plot.
id	The identifier for selecting the data frame source.

y_left	The variable used on the left y-axis when creating a comparative plot.
y_right	The variable used on the right y-axis when creating a comparative plot.
facet_var	The variable used for facet wrapping.
facet_name_var	The name of the variable used for facet wrapping.

Value

A list of ggplot objects.

`keep_only_relevant_values` *Keep only relevant values*

Description

Filters out only relevant values based on the provided filters

Usage

```
keep_only_relevant_values(lFilters, sVariable, dfFilters)
```

Arguments

lFilters	List of filters to be applied on the data.
sVariable	The variable for which relevant values are to be retrieved.
dfFilters	Dataframe with the possible filters and values for this dataset

Details

This function removes null elements from the filter list, transforms filter list into elements suitable for filtering, and retrieves relevant values from the data.

Value

A list of relevant values for the specified variable.

Examples

```
dfFilters <- dplyr::tibble(
  var1 = sample(c("A", "B", "C"), 100, replace = TRUE),
  var2 = sample(c("D", "E", "F"), 100, replace = TRUE),
  var3 = sample(c("G", "H", "I"), 100, replace = TRUE)
)
filters <- list("D;var2")
relevant_values <- keep_only_relevant_values(filters, "var1", dfFilters)

# Check if the relevant values are only from the rows where var2 is "D" or "E"
```

```
expected_values <- dfFilters$var1[dfFilters$var2 %in% c("D")] %>%
  purrr::set_names(.) %>%
  purrr::map(~paste0(.x, ";var1"))
```

---

keep_values *Keep values*

Description

This function extracts values before the semicolon from a ";"-separated string.

Usage

```
keep_values(input)
```

Arguments

input	A character vector with ";"-separated strings

Value

A list of values before the semicolon in the input

Examples

```
input = c("A;var1", "B;var1", "C;var1")
values = keep_values(input)
```

---

pickerGanttValues *pickerGanttValues function*

Description

Function to filter the values in the Gantt chart.

Usage

```
pickerGanttValues(id, filter_var, df_doorstroom_gantt)
```

Arguments

id	A string representing the id of the input element.
filter_var	A string representing the variable to filter.
df_doorstroom_gantt	A data frame containing the Gantt chart data.

Value

A pickerInput object.

pickerGanttVar *pickerGanttVar function*

Description

Function to pick a variable to show values in a Gantt chart.

Usage

```
pickerGanttVar(id, element, df_config_gantt, input_var_value = NULL)
```

Arguments

id	A string representing the id of the input element.
element	A string representing the element.
df_config_gantt	A data frame containing the Gantt configuration.
input_var_value	A variable value from the input. Default is NULL.

pickerSankeyValues *pickerSankeyValues function*

Description

Function to pick values for transition of the two Sankey states.

Usage

```
pickerSankeyValues(id, filter_var, df_sankey, side)
```

Arguments

id	A string representing the id of the input element.
filter_var	A string representing the variable to filter.
df_sankey	A data frame containing the Sankey diagram data.
side	A string representing the side of the Sankey diagram.

Value

A pickerInput object.

pickerSankeyVar *pickerSankeyVar function*

Description

Function to pick variables for state in a Sankey diagram.

Usage

```
pickerSankeyVar(id, df_sankey, df_config_sankey, state = "left_var")
```

Arguments

id	A string representing the id of the input element.
df_sankey	A data frame containing the Sankey diagram data.
df_config_sankey	A data frame containing the Sankey configuration.
state	A string representing the state of the variable. Default is "left_var".

Value

A pickerInput object.

pickerSplitVar *pickerSplitVar function*

Description

Function to create a picker input for splitting variables.

Usage

```
pickerSplitVar(
  id,
  variable = "INS_Splits_variabele",
  name = "color",
  label = "Kleur",
  df
)
```

Arguments

id	A string representing the id of the input element.
variable	A string representing the variable to split. Default is "INS_Splits_variabele".
name	A string representing the name. Default is "color".
label	A string representing the label of the input. Default is "Kleur".
df	A data frame containing the data. Default is dfCombi_geaggregeerd.

Value

A pickerInput object.

---

pickerValues *pickerValues function*

Description

Function to create a picker input for filtering value.

Usage

```
pickerValues(
  id,
  df,
  variable = "faculty",
  role = "left",
  selected = "All",
  multiple = TRUE
)
```

Arguments

id	A string representing the id of the input element.
df	A data frame containing the data.
variable	A string representing the variable to filter. Default is "faculty".
role	A string representing the role. Default is "left".
selected	The selected value. Default is "All".
multiple	A boolean indicating whether multiple selections are allowed. Default is TRUE.

Value

A pickerInput object.

pickerVar *pickerVar function*

Description

Function to generate a picker input element based on given id and element.

Usage

```
pickerVar(id, element, df_categories, label = NULL)
```

Arguments

id	A string representing the id of the input element.
element	A string representing the element.
df_categories	A dataframe with metadata about the available categories per picker element
label	A string representing the label of the input. Default is NULL.

Value

A pickerInput object.

prep_df *Prepare a dataframe*

Description

Prepares a dataframe based on provided filters and naming options

Usage

```
prep_df(
  lFilters,
  lValues_for_naming,
  df,
  color_var,
  facet = "left",
  facet_var = rlang::sym("VIS_Groep"),
  facet_name_var = rlang::sym("VIS_Groep_naam")
)
```

Arguments

lFilters	List of filters to be applied on the dataframe.
lValues_for_naming	List of values used for naming.
df	Dataframe to be processed.
color_var	Variable used for coloring.
facet	Facet grid side ("left" by default).
facet_var	Variable used for facet grid ("VIS_Groep" by default).
facet_name_var	Variable used for facet grid naming ("VIS_Groep_naam" by default).

Details

This function collapses values from the naming list into a single string, removes null elements from the filter list, transforms filter list into elements suitable for filtering, applies filters, adds new columns, and casts var used as color to factor.

Value

A prepared dataframe with applied filters and new columns.

Examples

```
df <- dplyr::tibble(
  VIS_Groep = sample(c("Group1", "Group2", "Group3"), 100, replace = TRUE),
  VIS_Groep_naam = sample(c("Name1", "Name2", "Name3"), 100, replace = TRUE),
  var1 = sample(c("A", "B", "C"), 100, replace = TRUE),
  var2 = rnorm(100),
  color_var = sample(c("Red", "Blue", "Green"), 100, replace = TRUE)
)
lFilters <- list("A;var1")
lValues_for_naming = list("Name1;VIS_Groep_naam", "Name2;VIS_Groep_naam")
color_var = "color_var"
dfPrepared <- prep_df(lFilters, lValues_for_naming, df, color_var, facet = "right")
```

---

prep_df_summ *Prepare summarized dataframe*

Description

This function prepares a summarized dataframe based on provided variables and a y-variable. The function groups the dataframe by the provided variables, summarizes the y-variable, and counts the number of observations per group.

Usage

```
prep_df_summ(df, variables, y)
```

Arguments

df	A dataframe to be summarized
variables	A character vector specifying the columns to be grouped by
y	A character vector specifying the column to be summarized

Value

A summarized dataframe

Examples

```
df <- data.frame(
id = c(1, 1, 2, 2),
group = c("A", "A", "B", "B"),
value = c(2, 4, 6, 8)
)
df_summ <- prep_df_summ(df, c("id", "group"), "value")
```

---

prep_df_summ_aggr *Prepare summarized and aggregated dataframe*

Description

This function prepares a summarized dataframe based on provided variables, y-variable, color, and total count. The function groups the dataframe by the provided variables, calculates the weighted mean for the y-variable, sums up total count per group, and arranges by color.

Usage

```
prep_df_summ_aggr(
  df,
  variables,
  y,
  color,
  total_n_var = rlang::sym("INS_Aantal_eerstejaars"),
  aggr_split_value_var = rlang::sym("INS_Splits_variabele_waarde")
)
```

Arguments

df	A dataframe to be summarized
variables	A character vector specifying the columns to be grouped by
y	A character vector specifying the column to be summarized
color	A character vector specifying the column to be used for color arrangement
total_n_var	A symbol specifying the variable to be used for total count calculation
aggr_split_value_var	A symbol specifying the variable to be used for color assignment

Value

A summarized and aggregated dataframe arranged by color

Examples

```
df <- data.frame( split_var_value = c("male", "male", "female", "female", "dutch", "dutch",
"EER", "EER", "Outside EER", "Outside EER"), other_var = c("Early", "Late", "Early", "Late",
"Early", "Late", "Early", "Late", "Early", "Late"), value = c(2, 4, 6, 8, 10, 2, 4, 6, 8, 10),
total = c(10, 10, 20, 20, 30, 30, 40, 40, 50, 50), split_var = c("gender", "gender", "gender",
"gender", "background", "background", "background", "background", "background", "background") )
```

---

prep_table *Prepare a data table for displaying*

Description

This function prepares a data table for displaying by providing user-friendly names, removing unneeded variables, and formatting percentages.

Usage

```
prep_table(
  y,
  df,
  df_summarized,
  id,
  y_right = NULL,
  facet_var = rlang::sym("VIS_Groep"),
  facet_name_var = rlang::sym("VIS_Groep_naam"),
  table_type = c("basic", "advanced"),
  rownames = FALSE,
  extensions = c("Buttons"),
  options_DT = basic_options(),
  limit_width = "values",
  ...
)
```

Arguments

y	A string specifying the column name to be used as the y-axis variable.
df	A data frame containing the raw data.
df_summarized	A data frame containing the summarized data.
id	A string specifying the ID associated with the data.
y_right	An optional string specifying the column name to be used as the second y-axis variable. Default is NULL.
facet_var	A symbol specifying the column to be used for faceting. Default is 'VIS_Groep'.

facet_name_var	A symbol specifying the column to be used for faceting names. Default is 'VIS_Groep_naam'.
table_type	Choose from basic for a simple datatable and advanced for more buttons etc.
rownames	A logical value indicating whether to display row names.
extensions	A character vector specifying the DataTables extensions to be used.
options_DT	A list of DataTables options.
limit_width	A character string indicating how to limit column width.
...	Further arguments passed on to the 'make_basic_table' function.

Value

A DT::datatable object ready for displaying.

Examples

```
df <- data.frame(VIS_Groep = c("Group1", "Group1", "Group2", "Group2"),
                 VIS_Groep_naam = c("Name1", "Name1", "Name2", "Name2"),
                 y = c(TRUE, TRUE, FALSE, FALSE), z = c(TRUE, FALSE, TRUE, FALSE))
df_summarized <- df %>%
 dplyr::group_by(VIS_Groep, VIS_Groep_naam) %>%
 dplyr::summarise(
   y = mean(y),
   z = mean(z)
 ) %>%
 dplyr::ungroup()
 id <- "id"
output <- prep_table("y", df, df_summarized, id = id)
```

---

present_and_correct *present_and_correct function*

Description

Function to check if the column is present and correctly formed based on the element type.

Usage

```
present_and_correct(column_name, element = NA, df)
```

Arguments

column_name	A string representing the column name.
element	A string representing the element. Default is NA.
df	A data frame for which to check the column. Default is dfCombi_geaggregeerd.

Value

A boolean indicating whether the column is present and correctly formed.

quietly_run *Quietly run a function*

Description

This function is a wrapper that allows a function to be run quietly without the need to create a separate quiet function.

Usage

```
quietly_run(func, ...)
```

Arguments

func	The function to be run.
...	Optional further arguments passed to the 'func' function.

Value

The list result of the 'func' function with messages, warnings, and output captured.

Examples

```
warning_func_arugment <- function(info) {
  warning(info)
  return("Complete")
}
result <- quietly_run(warning_func_arugment, "Just checking")
```

sankey_plot *Create a Sankey plot using ggplot and ggalluvial*

Description

This function creates a Sankey plot with the help of ggplot and ggalluvial.

Usage

```
sankey_plot(
  df,
  left_var,
  right_var,
  xlab_setting,
  ylab_setting,
  name_left,
  name_right,
```

```
  title,
  title_size = 20,
  title_font = "verdana"
)
```

Arguments

df	A data frame containing the data to be plotted.
left_var	A string specifying the column name to be used as the left variable.
right_var	A string specifying the column name to be used as the right variable.
xlab_setting	ggplot labels settings for x axes.
ylab_setting	ggplot labels settings for y axes.
name_left	A string specifying the name for the left side of the plot.
name_right	A string specifying the name for the right side of the plot.
title	A string specifying the title of the plot.
title_size	Numeric value specifying the size of the title.
title_font	A string specifying the font of the title.

Value

A Sankey plot.

---

single_module_ui *UI function for single module dashboard*

Description

UI function for single module dashboard

Usage

```
single_module_ui(request, id, tab_item)
```

Arguments

request	shiny request object
id	Module id
tab_item	Tab item UI

Value

dashboardPage UI

stacked_composition_bar_chart *Stacked bar chart*

Description

Create a stacked bar chart, with optional settings for percentage (or not) and wrap (or not) modes.

Usage

```
stacked_composition_bar_chart(
  df,
  x,
  color,
  id,
  facet_name_var = rlang::sym("VIS_Groep_naam"),
  percentage = FALSE,
  wrap = FALSE
)
```

Arguments

df	The data frame used to create the plot.
x	The variable used on the x-axis of the plot.
color	The variable used to color the points or bars in the plot.
id	The identifier for selecting the data frame source.
facet_name_var	The name of the variable used for facet wrapping.
percentage	Logical indicating whether to create a plot in percentage mode.
wrap	Logical indicating whether to use facet wrapping.

Value

A ggplot object.

tabTableOne *tabTableOne function*

Description

Function to create a tab panel with one table.

Usage

```
tabTableOne(id, table_one)
```

Arguments

id	A string representing the id.
table_one	A string representing the table.

Value

A tab panel with one table.

Examples

```
dummy_data <- data.frame(
  A = 1:5,
  B = letters[1:5]
)
dummy_dt <- DT::datatable(dummy_data)
tabTableOne("dummy_id", dummy_dt)
```

---

tabTableTwo *tabTableTwo function*

Description

Function to create a tab panel with two tables.

Usage

```
tabTableTwo(id, table_one, table_two)
```

Arguments

id	A string representing the id.
table_one	A string representing the first table.
table_two	A string representing the second table.

Value

A tab panel with two tables.

Examples

```
dummy_data1 <- data.frame(
  A = 1:5,
  B = letters[1:5]
)
dummy_dt1 <- DT::datatable(dummy_data1)
dummy_data2 <- data.frame(
X = 6:10,
Y = letters[6:10]
```

```
)
dummy_dt2 <- DT::datatable(dummy_data2)
tabTableTwo("dummy_id", dummy_dt1, dummy_dt2)
```

`taskItemTab` *taskItemTab function*

Description

Item for above dropdownActionMenu function.

Usage

```
taskItemTab(text, tab_name = NULL, href = NULL, tabSelect = FALSE)
```

Arguments

`text`	The text to display for the item.
`tab_name`	The name of the tab to link to. Default is NULL.
`href`	The href link for the item. If NULL, it defaults to "#".
`tabSelect`	A boolean indicating whether to select the tab. Default is FALSE.

Value

An HTML list item.

Examples

```
taskItemTab(text = "Selected tab", tab_name = "Tab1", tabSelect = TRUE)
taskItemTab(text = "Other tab", tab_name = "Tab2", tabSelect = FALSE)
```

`transform_input` *Transform input*

Description

This function transforms a list of inputs into a column and value for filtering.

Usage

```
transform_input(input)
```

Arguments

`input`	A list of inputs to be transformed

Value

A list containing a column and its corresponding value for filtering

Examples

```
input = list("A;var1", "B;var1", "C;var1")
filter_element = transform_input(input)
```

`wrapped_chart` *Wrapped chart*

Description

Wrapped chart function for creating a plot based on the provided dataframe and variables.

Usage

```
wrapped_chart(
  df,
  x,
  y,
  color,
  id = "bench",
  df_original,
  y_left = NULL,
  y_right = NULL,
  facet_var = rlang::sym("VIS_Groep"),
  facet_name_var = rlang::sym("VIS_Groep_naam")
)
```

Arguments

`df`	The data frame used to create the plot.
`x`	The variable used on the x-axis of the plot.
`y`	The variable used on the y-axis of the plot.
`color`	The variable used to color the points or bars in the plot.
`id`	The identifier for selecting the data frame source.
`df_original`	The original dataframe before summarization
`y_left`	The variable used on the left y-axis when creating a comparative plot.
`y_right`	The variable used on the right y-axis when creating a comparative plot.
`facet_var`	The variable used for facet wrapping.
`facet_name_var`	The name of the variable used for facet wrapping.

Value

A ggplot object.

Index