Package: teal.goshawk (via r-universe)

August 29, 2024

Type Package

Title Longitudinal Visualization `teal` Modules

Version 0.2.0

Date 2024-07-29

Description Modules that produce web interfaces through which longitudinal visualizations can be dynamically modified and displayed. These included box plot, correlation plot, density distribution plot, line plot, scatter plot and spaghetti plot with accompanying summary. Data are expected in ADaM structure. Requires analysis subject level (ADSL) and analysis laboratory (ADLB) data sets. Beyond core variables, Limit of Quantification flag variable (LOQFL) is expected with levels 'Y', 'N' or NA.

License Apache License 2.0 | file LICENSE

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

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

Depends goshawk (>= 0.1.18), R (>= 3.6), shiny (>= 1.6.0), teal (>= 0.15.2), teal.transform (>= 0.5.0)

Imports checkmate (>= 2.1.0), colourpicker, dplyr (>= 1.0.5), DT (>= 0.13), ggplot2 (>= 3.4.0), grDevices, lifecycle (>= 0.2.0), methods, rlang (>= 1.0.0), shinyjs, shinyvalidate, stats, teal.code (>= 0.5.0), teal.logger (>= 0.2.0), teal.reporter (>= 0.2.0), teal.widgets (>= 0.4.0)

Suggests knitr (>= 1.42), nestcolor (>= 0.1.0), rmarkdown (>= 2.23), stringr (>= 1.4.1), teal.data (>= 0.5.0), tern (>= 0.7.10), testthat (>= 3.0.4), utils

VignetteBuilder knitr

Config/Needs/verdepcheck insightsengineering/goshawk, rstudio/shiny, insightsengineering/teal, insightsengineering/teal.transform, mllg/checkmate, daattali/colourpicker, tidyverse/dplyr,

rstudio/DT, tidyverse/ggplot2, r-lib/lifecycle, r-lib/rlang, daattali/shinyjs, rstudio/shinyvalidate, insightsengineering/teal.code, insightsengineering/teal.logger, insightsengineering/teal.reporter, insightsengineering/teal.widgets, tidyverse/tidyr, yihui/knitr, insightsengineering/nestcolor, rstudio/rmarkdown, tidyverse/stringr, insightsengineering/teal.data, insightsengineering/tern, r-lib/testthat

Config/Needs/website insightsengineering/nesttemplate

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://insightsengineering.r-universe.dev

RemoteUrl https://github.com/insightsengineering/teal.goshawk

RemoteRef v0.2.0

RemoteSha 3739198bbdde0ca0b40f89c1e1eab685ecc50799

Contents

`maptrt` *helper for writing arm mapping and ordering code.*

Description

Provides lines of code for left hand side of arm mapping. user must provide right hand side

Usage

```
maptrt(df_armvar, code = c("M", "O"))
```

Arguments

df_armvar	the dataframe and column name containing treatment code. e.g. ADSL$ARMCD
code	controls whether mapping or ordering code is written to console. Valid values: "M" and "O".

Details

SPA configure study specific pre-processing for deploying goshawk. writing the code for ARM mapping and ordering is tedious. this function helps to get that started by providing the left hand side of the mapping and ordering syntax. call the function and then copy and paste the resulting code from the console into the app.R file.

Examples

```
ADSL <- rADSL

# get treatment mapping code
maptrt(df_armvar = ADSL$ARMCD, code = "M")

# get treatment ordering code
maptrt(df_armvar = ADSL$ARMCD, code = "O")
```

---

tm_g_gh_boxplot *Box Plot*

Description

This teal module renders the UI and calls the functions that create a box plot and accompanying summary table.

Usage

```
tm_g_gh_boxplot(
  label,
  dataname,
  param_var,
  param,
  yaxis_var = teal.transform::choices_selected(c("AVAL", "CHG"), "AVAL"),
  xaxis_var = teal.transform::choices_selected("AVISITCD", "AVISITCD"),
  facet_var = teal.transform::choices_selected(c("ARM", "ACTARM"), "ARM"),
  trt_group,
  color_manual = NULL,
  shape_manual = NULL,
  facet_ncol = NULL,
  loq_legend = TRUE,
  rotate_xlab = FALSE,
  hline_arb = numeric(0),
```

```
  hline_arb_color = "red",
  hline_arb_label = "Horizontal line",
  hline_vars = character(0),
  hline_vars_colors = "green",
  hline_vars_labels = hline_vars,
  plot_height = c(600, 200, 2000),
  plot_width = NULL,
  font_size = c(12, 8, 20),
  dot_size = c(2, 1, 12),
  alpha = c(0.8, 0, 1),
  pre_output = NULL,
  post_output = NULL
)
```

Arguments

label	menu item label of the module in the teal app.
dataname	analysis data passed to the data argument of init. E.g. ADaM structured laboratory data frame ALB.
param_var	name of variable containing biomarker codes e.g. PARAMCD.
param	list of biomarkers of interest.
yaxis_var	name of variable containing biomarker results displayed on y-axis e.g. AVAL. When not provided, it defaults to choices_selected(c("AVAL", "CHG"), "AVAL").
xaxis_var	variable to categorize the x-axis. When not provided, it defaults to choices_selected("AVISITCD", "AVISITCD").
facet_var	variable to facet the plots by. When not provided, it defaults to choices_selected(c("ARM", "ACTARM"), "ARM").
trt_group	choices_selected object with available choices and pre-selected option for variable names representing treatment group e.g. ARM.
color_manual	vector of colors applied to treatment values.
shape_manual	vector of symbols applied to LOQ values.
facet_ncol	numeric value indicating number of facets per row.
loq_legend	loq legend toggle.
rotate_xlab	45 degree rotation of x-axis values.
hline_arb	numeric vector of at most 2 values identifying intercepts for arbitrary horizontal lines.
hline_arb_color	a character vector of at most length of hline_arb. naming the color for the arbitrary horizontal lines.
hline_arb_label	a character vector of at most length of hline_arb. naming the label for the arbitrary horizontal lines.
hline_vars	a character vector to name the columns that will define additional horizontal lines.

hline_vars_colors
: a character vector naming the colors for the additional horizontal lines.

hline_vars_labels
: a character vector naming the labels for the additional horizontal lines that will appear in the legend.

plot_height
: controls plot height.

plot_width
: optional, controls plot width.

font_size
: font size control for title, `x-axis` label, `y-axis` label and legend.

dot_size
: plot dot size.

alpha
: numeric vector to define transparency of plotted points.

pre_output
: (`shiny.tag`) optional,
with text placed before the output to put the output into context. For example a title.

post_output
: (`shiny.tag`) optional, with text placed after the output to put the output into context. For example the `shiny::helpText()` elements are useful.

Value

an `module` object

Author(s)

Jeff Tomlinson (tomlinsj) jeffrey.tomlinson@roche.com

Balazs Toth (tothb2) toth.balazs@gene.com

Examples

```
# Example using ADaM structure analysis dataset.
data <- teal_data()
data <- within(data, {
  library(dplyr)
  library(nestcolor)
  library(stringr)

  # use non-exported function from goshawk
  h_identify_loq_values <- getFromNamespace("h_identify_loq_values", "goshawk")

  # original ARM value = dose value
  arm_mapping <- list(
    "A: Drug X" = "150mg QD",
    "B: Placebo" = "Placebo",
    "C: Combination" = "Combination"
  )
  set.seed(1)
  ADSL <- rADSL
  ADLB <- rADLB
  var_labels <- lapply(ADLB, function(x) attributes(x)$label)
  ADLB <- ADLB %>%
```

```
    mutate(
      AVISITCD = case_when(
        AVISIT == "SCREENING" ~ "SCR",
        AVISIT == "BASELINE" ~ "BL",
        grepl("WEEK", AVISIT) ~ paste("W", str_extract(AVISIT, "(?<=(WEEK ))[0-9]+")),
        TRUE ~ as.character(NA)
      ),
      AVISITCDN = case_when(
        AVISITCD == "SCR" ~ -2,
        AVISITCD == "BL" ~ 0,
        grepl("W", AVISITCD) ~ as.numeric(gsub("[^0-9]*", "", AVISITCD)),
        TRUE ~ as.numeric(NA)
      ),
      AVISITCD = factor(AVISITCD) %>% reorder(AVISITCDN),
      TRTORD = case_when(
        ARMCD == "ARM C" ~ 1,
        ARMCD == "ARM B" ~ 2,
        ARMCD == "ARM A" ~ 3
      ),
      ARM = as.character(arm_mapping[match(ARM, names(arm_mapping))]),
      ARM = factor(ARM) %>% reorder(TRTORD),
      ACTARM = as.character(arm_mapping[match(ACTARM, names(arm_mapping))]),
      ACTARM = factor(ACTARM) %>% reorder(TRTORD),
      ANRLO = 50,
      ANRHI = 75
    ) %>%
    rowwise() %>%
    group_by(PARAMCD) %>%
    mutate(LBSTRESC = ifelse(
      USUBJID %in% sample(USUBJID, 1, replace = TRUE),
      paste("<", round(runif(1, min = 25, max = 30))), LBSTRESC
    )) %>%
    mutate(LBSTRESC = ifelse(
      USUBJID %in% sample(USUBJID, 1, replace = TRUE),
      paste(">", round(runif(1, min = 70, max = 75))), LBSTRESC
    )) %>%
    ungroup()

  attr(ADLB[["ARM"]], "label") <- var_labels[["ARM"]]
  attr(ADLB[["ACTARM"]], "label") <- var_labels[["ACTARM"]]
  attr(ADLB[["ANRLO"]], "label") <- "Analysis Normal Range Lower Limit"
  attr(ADLB[["ANRHI"]], "label") <- "Analysis Normal Range Upper Limit"

  # add LLOQ and ULOQ variables
  ALB_LOQS <- h_identify_loq_values(ADLB, "LOQFL")
  ADLB <- left_join(ADLB, ALB_LOQS, by = "PARAM")
})

datanames <- c("ADSL", "ADLB")
datanames(data) <- datanames

join_keys(data) <- default_cdisc_join_keys[datanames]
```

```
app <- init(
  data = data,
  modules = modules(
    tm_g_gh_boxplot(
      label = "Box Plot",
      dataname = "ADLB",
      param_var = "PARAMCD",
      param = choices_selected(c("ALT", "CRP", "IGA"), "ALT"),
      yaxis_var = choices_selected(c("AVAL", "BASE", "CHG"), "AVAL"),
      xaxis_var = choices_selected(c("ACTARM", "ARM", "AVISITCD", "STUDYID"), "ARM"),
      facet_var = choices_selected(c("ACTARM", "ARM", "AVISITCD", "SEX"), "AVISITCD"),
      trt_group = choices_selected(c("ARM", "ACTARM"), "ARM"),
      loq_legend = TRUE,
      rotate_xlab = FALSE,
      hline_arb = c(60, 55),
      hline_arb_color = c("grey", "red"),
      hline_arb_label = c("default_hori_A", "default_hori_B"),
      hline_vars = c("ANRHI", "ANRLO", "ULOQN", "LLOQN"),
      hline_vars_colors = c("pink", "brown", "purple", "black"),
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_g_gh_correlationplot` *Scatter Plot Teal Module For Biomarker Analysis*

Description

Scatter Plot Teal Module For Biomarker Analysis

Usage

```
tm_g_gh_correlationplot(
  label,
  dataname,
  param_var = "PARAMCD",
  xaxis_param = "ALT",
  xaxis_var = "BASE",
  yaxis_param = "CRP",
  yaxis_var = "AVAL",
  trt_group,
  color_manual = NULL,
  shape_manual = NULL,
  facet_ncol = 2,
```

```
  visit_facet = TRUE,
  trt_facet = FALSE,
  reg_line = FALSE,
  loq_legend = TRUE,
  rotate_xlab = FALSE,
  hline_arb = numeric(0),
  hline_arb_color = "red",
  hline_arb_label = "Horizontal line",
  hline_vars = character(0),
  hline_vars_colors = "green",
  hline_vars_labels = hline_vars,
  vline_arb = numeric(0),
  vline_arb_color = "red",
  vline_arb_label = "Vertical line",
  vline_vars = character(0),
  vline_vars_colors = "green",
  vline_vars_labels = vline_vars,
  plot_height = c(500, 200, 2000),
  plot_width = NULL,
  font_size = c(12, 8, 20),
  dot_size = c(1, 1, 12),
  reg_text_size = c(3, 3, 10),
  pre_output = NULL,
  post_output = NULL
)
```

Arguments

label	menu item label of the module in the teal app.
dataname	analysis data passed to the data argument of init. E.g. ADaM structured laboratory data frame ADLB.
param_var	name of variable containing biomarker codes e.g. PARAMCD.
xaxis_param	biomarker selected for x-axis.
xaxis_var	name of variable containing biomarker results displayed on x-axis e.g. BASE.
yaxis_param	biomarker selected for y-axis.
yaxis_var	name of variable containing biomarker results displayed on y-axis e.g. AVAL.
trt_group	choices_selected object with available choices and pre-selected option for variable names representing treatment group e.g. ARM.
color_manual	vector of colors applied to treatment values.
shape_manual	vector of symbols applied to LOQ values.
facet_ncol	numeric value indicating number of facets per row.
visit_facet	visit facet toggle.
trt_facet	facet by treatment group trt_group.
reg_line	include regression line and annotations for slope and coefficient in visualization. Use with facet TRUE.

loq_legend	loq legend toggle.
rotate_xlab	45 degree rotation of x-axis values.
hline_arb	numeric vector of at most 2 values identifying intercepts for arbitrary horizontal lines.
hline_arb_color	a character vector of at most length of hline_arb. naming the color for the arbitrary horizontal lines.
hline_arb_label	a character vector of at most length of hline_arb. naming the label for the arbitrary horizontal lines.
hline_vars	a character vector to name the columns that will define additional horizontal lines.
hline_vars_colors	a character vector naming the colors for the additional horizontal lines.
hline_vars_labels	a character vector naming the labels for the additional horizontal lines that will appear
vline_arb	numeric vector of at most 2 values identifying intercepts for arbitrary horizontal lines.
vline_arb_color	a character vector of at most length of vline_arb. naming the color for the arbitrary horizontal lines.
vline_arb_label	a character vector of at most length of vline_arb. naming the label for the arbitrary horizontal lines.
vline_vars	a character vector to name the columns that will define additional vertical lines.
vline_vars_colors	a character vector naming the colors for the additional vertical lines.
vline_vars_labels	a character vector naming the labels for the additional vertical lines that will appear
plot_height	controls plot height.
plot_width	optional, controls plot width.
font_size	font size control for title, x-axis label, y-axis label and legend.
dot_size	plot dot size.
reg_text_size	font size control for regression line annotations.
pre_output	(shiny.tag) optional, with text placed before the output to put the output into context. For example a title.
post_output	(shiny.tag) optional, with text placed after the output to put the output into context. For example the shiny::helpText() elements are useful.

Author(s)

Nick Paszty (npaszty) paszty.nicholas@gene.com

Balazs Toth (tothb2) toth.balazs@gene.com

Examples

```
# Example using ADaM structure analysis dataset.
data <- teal_data()
data <- within(data, {
  library(dplyr)
  library(stringr)

  # use non-exported function from goshawk
  h_identify_loq_values <- getFromNamespace("h_identify_loq_values", "goshawk")

  # original ARM value = dose value
  arm_mapping <- list(
    "A: Drug X" = "150mg QD",
    "B: Placebo" = "Placebo",
    "C: Combination" = "Combination"
  )
 color_manual <- c("150mg QD" = "#000000", "Placebo" = "#3498DB", "Combination" = "#E74C3C")
  # assign LOQ flag symbols: circles for "N" and triangles for "Y", squares for "NA"
  shape_manual <- c("N" = 1, "Y" = 2, "NA" = 0)

  set.seed(1)
  ADSL <- rADSL
  ADLB <- rADLB
  var_labels <- lapply(ADLB, function(x) attributes(x)$label)
  ADLB <- ADLB %>%
    mutate(AVISITCD = case_when(
      AVISIT == "SCREENING" ~ "SCR",
      AVISIT == "BASELINE" ~ "BL",
      grepl("WEEK", AVISIT) ~
        paste(
          "W",
          trimws(
            substr(
              AVISIT,
              start = 6,
              stop = str_locate(AVISIT, "DAY") - 1
            )
          )
        ),
      TRUE ~ NA_character_
    )) %>%
    mutate(AVISITCDN = case_when(
      AVISITCD == "SCR" ~ -2,
      AVISITCD == "BL" ~ 0,
      grepl("W", AVISITCD) ~ as.numeric(gsub("[^0-9]*", "", AVISITCD)),
      TRUE ~ NA_real_
    )) %>%
    # use ARMCD values to order treatment in visualization legend
    mutate(TRTORD = ifelse(grepl("C", ARMCD), 1,
      ifelse(grepl("B", ARMCD), 2,
        ifelse(grepl("A", ARMCD), 3, NA)
      )
```

```
    )) %>%
    mutate(ARM = as.character(arm_mapping[match(ARM, names(arm_mapping))])) %>%
    mutate(ARM = factor(ARM) %>%
      reorder(TRTORD)) %>%
    mutate(
      ANRHI = case_when(
        PARAMCD == "ALT" ~ 60,
        PARAMCD == "CRP" ~ 70,
        PARAMCD == "IGA" ~ 80,
        TRUE ~ NA_real_
      ),
      ANRLO = case_when(
        PARAMCD == "ALT" ~ 20,
        PARAMCD == "CRP" ~ 30,
        PARAMCD == "IGA" ~ 40,
        TRUE ~ NA_real_
      )
    ) %>%
    rowwise() %>%
    group_by(PARAMCD) %>%
    mutate(LBSTRESC = ifelse(
      USUBJID %in% sample(USUBJID, 1, replace = TRUE),
      paste("<", round(runif(1, min = 25, max = 30))), LBSTRESC
    )) %>%
    mutate(LBSTRESC = ifelse(
      USUBJID %in% sample(USUBJID, 1, replace = TRUE),
      paste(">", round(runif(1, min = 70, max = 75))), LBSTRESC
    )) %>%
    ungroup()
  attr(ADLB[["ARM"]], "label") <- var_labels[["ARM"]]
  attr(ADLB[["ANRHI"]], "label") <- "Analysis Normal Range Upper Limit"
  attr(ADLB[["ANRLO"]], "label") <- "Analysis Normal Range Lower Limit"

  # add LLOQ and ULOQ variables
  ADLB_LOQS <- h_identify_loq_values(ADLB, "LOQFL")
  ADLB <- left_join(ADLB, ADLB_LOQS, by = "PARAM")
})

datanames <- c("ADSL", "ADLB")
datanames(data) <- datanames

join_keys(data) <- default_cdisc_join_keys[datanames]

app <- init(
  data = data,
  modules = modules(
    tm_g_gh_correlationplot(
      label = "Correlation Plot",
      dataname = "ADLB",
      param_var = "PARAMCD",
      xaxis_param = choices_selected(c("ALT", "CRP", "IGA"), "ALT"),
      yaxis_param = choices_selected(c("ALT", "CRP", "IGA"), "CRP"),
      xaxis_var = choices_selected(c("AVAL", "BASE", "CHG", "PCHG"), "BASE"),
```

```
      yaxis_var = choices_selected(c("AVAL", "BASE", "CHG", "PCHG"), "AVAL"),
      trt_group = choices_selected(c("ARM", "ACTARM"), "ARM"),
      color_manual = c(
        "Drug X 100mg" = "#000000",
        "Placebo" = "#3498DB",
        "Combination 100mg" = "#E74C3C"
      ),
      shape_manual = c("N" = 1, "Y" = 2, "NA" = 0),
      plot_height = c(500, 200, 2000),
      facet_ncol = 2,
      visit_facet = TRUE,
      reg_line = FALSE,
      loq_legend = TRUE,
      font_size = c(12, 8, 20),
      dot_size = c(1, 1, 12),
      reg_text_size = c(3, 3, 10),
      hline_arb = c(40, 50),
      hline_arb_label = "arb hori label",
      hline_arb_color = c("red", "blue"),
      hline_vars = c("ANRHI", "ANRLO", "ULOQN", "LLOQN"),
      hline_vars_colors = c("green", "blue", "purple", "cyan"),
      hline_vars_labels = c("ANRHI Label", "ANRLO Label", "ULOQN Label", "LLOQN Label"),
      vline_vars = c("ANRHI", "ANRLO", "ULOQN", "LLOQN"),
      vline_vars_colors = c("yellow", "orange", "brown", "gold"),
      vline_vars_labels = c("ANRHI Label", "ANRLO Label", "ULOQN Label", "LLOQN Label"),
      vline_arb = c(50, 70),
      vline_arb_label = "arb vert A",
      vline_arb_color = c("green", "orange")
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_g_gh_density_distribution_plot` *Density Distribution Plot*

Description

This teal module renders the UI and calls the functions that create a density distribution plot and an accompanying summary table.

Usage

```
tm_g_gh_density_distribution_plot(
  label,
  dataname,
```

```
  param_var,
  param,
  xaxis_var,
  trt_group,
  color_manual = NULL,
  color_comb = NULL,
  plot_height = c(500, 200, 2000),
  plot_width = NULL,
  font_size = c(12, 8, 20),
  line_size = c(1, 0.25, 3),
  hline_arb = numeric(0),
  hline_arb_color = "red",
  hline_arb_label = "Horizontal line",
  facet_ncol = 2L,
  comb_line = TRUE,
  rotate_xlab = FALSE,
  pre_output = NULL,
  post_output = NULL
)
```

Arguments

label	menu item label of the module in the teal app.
dataname	analysis data passed to the data argument of init. E.g. ADaM structured laboratory data frame ADLB.
param_var	name of variable containing biomarker codes e.g. PARAMCD.
param	biomarker selected.
xaxis_var	name of variable containing biomarker results displayed on x-axis e.g. BASE.
trt_group	choices_selected object with available choices and pre-selected option for variable names representing treatment group e.g. ARM.
color_manual	vector of colors applied to treatment values.
color_comb	name or hex value for combined treatment color.
plot_height	controls plot height.
plot_width	optional, controls plot width.
font_size	font size control for title, x-axis label, y-axis label and legend.
line_size	plot line thickness.
hline_arb	numeric vector of at most 2 values identifying intercepts for arbitrary horizontal lines.
hline_arb_color	a character vector of at most length of hline_arb. naming the color for the arbitrary horizontal lines.
hline_arb_label	a character vector of at most length of hline_arb. naming the label for the arbitrary horizontal lines.
facet_ncol	numeric value indicating number of facets per row.

comb_line	display combined treatment line toggle.
rotate_xlab	45 degree rotation of x-axis values.
pre_output	(shiny.tag) optional, with text placed before the output to put the output into context. For example a title.
post_output	(shiny.tag) optional, with text placed after the output to put the output into context. For example the shiny::helpText() elements are useful.

Details

None

Author(s)

Nick Paszty (npaszty) paszty.nicholas@gene.com

Balazs Toth (tothb2) toth.balazs@gene.com

Examples

```
# Example using ADaM structure analysis dataset.
data <- teal_data()
data <- within(data, {
  library(dplyr)
  library(stringr)

  # original ARM value = dose value
  arm_mapping <- list(
    "A: Drug X" = "150mg QD",
    "B: Placebo" = "Placebo",
    "C: Combination" = "Combination"
  )
  ADSL <- rADSL
  ADLB <- rADLB
  var_labels <- lapply(ADLB, function(x) attributes(x)$label)
  ADLB <- ADLB %>%
    mutate(
      AVISITCD = case_when(
        AVISIT == "SCREENING" ~ "SCR",
        AVISIT == "BASELINE" ~ "BL",
        grepl("WEEK", AVISIT) ~ paste("W", str_extract(AVISIT, "(?<=(WEEK ))[0-9]+")),
        TRUE ~ as.character(NA)
      ),
      AVISITCDN = case_when(
        AVISITCD == "SCR" ~ -2,
        AVISITCD == "BL" ~ 0,
        grepl("W", AVISITCD) ~ as.numeric(gsub("[^0-9]*", "", AVISITCD)),
        TRUE ~ as.numeric(NA)
      ),
      AVISITCD = factor(AVISITCD) %>% reorder(AVISITCDN),
      TRTORD = case_when(
        ARMCD == "ARM C" ~ 1,
```

```
          ARMCD == "ARM B" ~ 2,
          ARMCD == "ARM A" ~ 3
        ),
        ARM = as.character(arm_mapping[match(ARM, names(arm_mapping))]),
        ARM = factor(ARM) %>% reorder(TRTORD),
        ACTARM = as.character(arm_mapping[match(ACTARM, names(arm_mapping))]),
        ACTARM = factor(ACTARM) %>% reorder(TRTORD)
      )

    attr(ADLB[["ARM"]], "label") <- var_labels[["ARM"]]
    attr(ADLB[["ACTARM"]], "label") <- var_labels[["ACTARM"]]
  })

  datanames <- c("ADSL", "ADLB")
  datanames(data) <- datanames
  join_keys(data) <- default_cdisc_join_keys[datanames]

  app <- init(
    data = data,
    modules = modules(
      tm_g_gh_density_distribution_plot(
        label = "Density Distribution Plot",
        dataname = "ADLB",
        param_var = "PARAMCD",
        param = choices_selected(c("ALT", "CRP", "IGA"), "ALT"),
        xaxis_var = choices_selected(c("AVAL", "BASE", "CHG", "PCHG"), "AVAL"),
        trt_group = choices_selected(c("ARM", "ACTARM"), "ARM"),
        color_manual = c(
          "150mg QD" = "#000000",
          "Placebo" = "#3498DB",
          "Combination" = "#E74C3C"
        ),
        color_comb = "#39ff14",
        comb_line = TRUE,
        plot_height = c(500, 200, 2000),
        font_size = c(12, 8, 20),
        line_size = c(1, .25, 3),
        hline_arb = c(.02, .05),
        hline_arb_color = c("red", "black"),
        hline_arb_label = c("Horizontal Line A", "Horizontal Line B")
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

---

tm_g_gh_lineplot *Line plot*

Description

This teal module renders the UI and calls the function that creates a line plot.

Usage

```
tm_g_gh_lineplot(
  label,
  dataname,
  param_var,
  param,
  param_var_label = "PARAM",
  xaxis_var,
  yaxis_var,
  xvar_level = NULL,
  filter_var = yaxis_var,
  filter_var_choices = filter_var,
  trt_group,
  trt_group_level = NULL,
  shape_choices = NULL,
  stat = "mean",
  hline_arb = numeric(0),
  hline_arb_color = "red",
  hline_arb_label = "Horizontal line",
  color_manual = c(getOption("ggplot2.discrete.colour"), c("#ff0000", "#008000",
    "#4ca3dd", "#8a2be2"))[1:4],
  xtick = ggplot2::waiver(),
  xlabel = xtick,
  rotate_xlab = FALSE,
  plot_height = c(600, 200, 4000),
  plot_width = NULL,
  plot_font_size = c(12, 8, 20),
  dodge = c(0.4, 0, 1),
  pre_output = NULL,
  post_output = NULL,
  count_threshold = 0,
  table_font_size = c(12, 4, 20),
  dot_size = c(2, 1, 12),
  plot_relative_height_value = 1000
)
```

Arguments

label	menu item label of the module in the teal app.
dataname	analysis data passed to the data argument of init. E.g. ADaM structured laboratory data frame ADLB.
param_var	name of variable containing biomarker codes e.g. PARAMCD.
param	biomarker selected.

param_var_label
: single name of variable in analysis data that includes parameter labels.

xaxis_var
: single name of variable in analysis data that is used as x-axis in the plot for the respective goshawk function.

yaxis_var
: single name of variable in analysis data that is used as summary variable in the respective goshawk function.

xvar_level
: vector that can be used to define the factor level of xvar. Only use it when xvar is character or factor.

filter_var
: data constraint variable.

filter_var_choices
: data constraint variable choices.

trt_group
: choices_selected object with available choices and pre-selected option for variable names representing treatment group e.g. ARM.

trt_group_level
: vector that can be used to define factor level of trt_group.

shape_choices
: Vector or choices_selected object with names of ADSL variables which can be used to change shape

stat
: string of statistics

hline_arb
: numeric vector of at most 2 values identifying intercepts for arbitrary horizontal lines.

hline_arb_color
: a character vector of at most length of hline_arb. naming the color for the arbitrary horizontal lines.

hline_arb_label
: a character vector of at most length of hline_arb. naming the label for the arbitrary horizontal lines.

color_manual
: string vector representing customized colors

xtick
: numeric vector to define the tick values of x-axis when x variable is numeric. Default value is waive().

xlabel
: vector with same length of xtick to define the label of x-axis tick values. Default value is waive().

rotate_xlab
: logical(1) value indicating whether to rotate x-axis labels.

plot_height
: controls plot height.

plot_width
: optional, controls plot width.

plot_font_size
: control font size for title, x-axis, y-axis and legend font.

dodge
: controls the position dodge of error bar

pre_output
: (shiny.tag) optional,
with text placed before the output to put the output into context. For example a title.

post_output
: (shiny.tag) optional, with text placed after the output to put the output into context. For example the shiny::helpText() elements are useful.

`count_threshold`
: minimum count of observations (as listed in the output table) to plot nodes on the graph

`table_font_size`
: controls the font size of values in the table

`dot_size`
: plot dot size.

`plot_relative_height_value`
: numeric value between 500 and 5000 for controlling the starting value of the relative plot height slider

Value

`shiny` object

Author(s)

Wenyi Liu (luiw2) wenyi.liu@roche.com

Balazs Toth (tothb2) toth.balazs@gene.com

Examples

```
# Example using ADaM structure analysis dataset.
data <- teal_data()
data <- within(data, {
  library(dplyr)
  library(stringr)
  library(nestcolor)

  # original ARM value = dose value
  arm_mapping <- list(
    "A: Drug X" = "150mg QD",
    "B: Placebo" = "Placebo",
    "C: Combination" = "Combination"
  )

  ADSL <- rADSL
  ADLB <- rADLB
  var_labels <- lapply(ADLB, function(x) attributes(x)$label)
  ADLB <- ADLB %>%
    mutate(
      AVISITCD = case_when(
        AVISIT == "SCREENING" ~ "SCR",
        AVISIT == "BASELINE" ~ "BL",
        grepl("WEEK", AVISIT) ~ paste("W", str_extract(AVISIT, "(?<=(WEEK ))[0-9]+")),
        TRUE ~ as.character(NA)
      ),
      AVISITCDN = case_when(
        AVISITCD == "SCR" ~ -2,
        AVISITCD == "BL" ~ 0,
        grepl("W", AVISITCD) ~ as.numeric(gsub("[^0-9]*", "", AVISITCD)),
        TRUE ~ as.numeric(NA)
```

```
      ),
      AVISITCD = factor(AVISITCD) %>% reorder(AVISITCDN),
      TRTORD = case_when(
        ARMCD == "ARM C" ~ 1,
        ARMCD == "ARM B" ~ 2,
        ARMCD == "ARM A" ~ 3
      ),
      ARM = as.character(arm_mapping[match(ARM, names(arm_mapping))]),
      ARM = factor(ARM) %>% reorder(TRTORD),
      ACTARM = as.character(arm_mapping[match(ACTARM, names(arm_mapping))]),
      ACTARM = factor(ACTARM) %>% reorder(TRTORD)
    )
  attr(ADLB[["ARM"]], "label") <- var_labels[["ARM"]]
  attr(ADLB[["ACTARM"]], "label") <- var_labels[["ACTARM"]]
})

datanames <- c("ADSL", "ADLB")
datanames(data) <- datanames
join_keys(data) <- default_cdisc_join_keys[datanames]

app <- init(
  data = data,
  modules = modules(
    tm_g_gh_lineplot(
      label = "Line Plot",
      dataname = "ADLB",
      param_var = "PARAMCD",
      param = choices_selected(c("ALT", "CRP", "IGA"), "ALT"),
      shape_choices = c("SEX", "RACE"),
      xaxis_var = choices_selected("AVISITCD", "AVISITCD"),
      yaxis_var = choices_selected(c("AVAL", "BASE", "CHG", "PCHG"), "AVAL"),
      trt_group = choices_selected(c("ARM", "ACTARM"), "ARM"),
      hline_arb = c(20.5, 19.5),
      hline_arb_color = c("red", "green"),
      hline_arb_label = c("A", "B")
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_g_gh_scatterplot` *Scatter Plot Teal Module For Biomarker Analysis*

Description

[Deprecated]

`tm_g_gh_scatterplot` is deprecated. Please use tm_g_gh_correlationplot instead.

Usage

```
tm_g_gh_scatterplot(
  label,
  dataname,
  param_var,
  param,
  xaxis_var,
  yaxis_var,
  trt_group,
  color_manual = NULL,
  shape_manual = NULL,
  facet_ncol = 2,
  trt_facet = FALSE,
  reg_line = FALSE,
  rotate_xlab = FALSE,
  hline = NULL,
  vline = NULL,
  plot_height = c(500, 200, 2000),
  plot_width = NULL,
  font_size = c(12, 8, 20),
  dot_size = c(1, 1, 12),
  reg_text_size = c(3, 3, 10),
  pre_output = NULL,
  post_output = NULL
)
```

Arguments

label	menu item label of the module in the teal app.
dataname	analysis data passed to the data argument of init. E.g. ADaM structured laboratory data frame ADLB.
param_var	name of variable containing biomarker codes e.g. PARAMCD.
param	biomarker selected.
xaxis_var	name of variable containing biomarker results displayed on x-axis e.g. BASE.
yaxis_var	name of variable containing biomarker results displayed on y-axis e.g. AVAL.
trt_group	choices_selected object with available choices and pre-selected option for variable names representing treatment group e.g. ARM.
color_manual	vector of colors applied to treatment values.
shape_manual	vector of symbols applied to LOQ values.
facet_ncol	numeric value indicating number of facets per row.
trt_facet	facet by treatment group trt_group.
reg_line	include regression line and annotations for slope and coefficient in visualization. Use with facet TRUE.
rotate_xlab	45 degree rotation of x-axis values.

hline	y-axis value to position of horizontal line.
vline	x-axis value to position a vertical line.
plot_height	controls plot height.
plot_width	optional, controls plot width.
font_size	font size control for title, x-axis label, y-axis label and legend.
dot_size	plot dot size.
reg_text_size	font size control for regression line annotations.
pre_output	(shiny.tag) optional, with text placed before the output to put the output into context. For example a title.
post_output	(shiny.tag) optional, with text placed after the output to put the output into context. For example the shiny::helpText() elements are useful.

Author(s)

Nick Paszty (npaszty) paszty.nicholas@gene.com

Balazs Toth (tothb2) toth.balazs@gene.com

Examples

```
# Example using ADaM structure analysis dataset.
data <- teal_data()
data <- within(data, {
  library(dplyr)
  library(stringr)

  # original ARM value = dose value
  arm_mapping <- list(
    "A: Drug X" = "150mg QD",
    "B: Placebo" = "Placebo",
    "C: Combination" = "Combination"
  )

  ADSL <- rADSL
  ADLB <- rADLB
  var_labels <- lapply(ADLB, function(x) attributes(x)$label)
  ADLB <- ADLB %>%
    mutate(
      AVISITCD = case_when(
        AVISIT == "SCREENING" ~ "SCR",
        AVISIT == "BASELINE" ~ "BL",
        grepl("WEEK", AVISIT) ~ paste("W", str_extract(AVISIT, "(?<=(WEEK ))[0-9]+")),
        TRUE ~ as.character(NA)
      ),
      AVISITCDN = case_when(
        AVISITCD == "SCR" ~ -2,
        AVISITCD == "BL" ~ 0,
        grepl("W", AVISITCD) ~ as.numeric(gsub("[^0-9]*", "", AVISITCD)),
        TRUE ~ as.numeric(NA)
```

```
      ),
      AVISITCD = factor(AVISITCD) %>% reorder(AVISITCDN),
      TRTORD = case_when(
        ARMCD == "ARM C" ~ 1,
        ARMCD == "ARM B" ~ 2,
        ARMCD == "ARM A" ~ 3
      ),
      ARM = as.character(arm_mapping[match(ARM, names(arm_mapping))]),
      ARM = factor(ARM) %>% reorder(TRTORD),
      ACTARM = as.character(arm_mapping[match(ACTARM, names(arm_mapping))]),
      ACTARM = factor(ACTARM) %>% reorder(TRTORD)
    )
  attr(ADLB[["ARM"]], "label") <- var_labels[["ARM"]]
  attr(ADLB[["ACTARM"]], "label") <- var_labels[["ACTARM"]]
})

datanames <- c("ADSL", "ADLB")
datanames(data) <- datanames
join_keys(data) <- default_cdisc_join_keys[datanames]


app <- init(
  data = data,
  modules = modules(
    tm_g_gh_scatterplot(
      label = "Scatter Plot",
      dataname = "ADLB",
      param_var = "PARAMCD",
      param = choices_selected(c("ALT", "CRP", "IGA"), "ALT"),
      xaxis_var = choices_selected(c("AVAL", "BASE", "CHG", "PCHG"), "BASE"),
      yaxis_var = choices_selected(c("AVAL", "BASE", "CHG", "PCHG"), "AVAL"),
      trt_group = choices_selected(c("ARM", "ACTARM"), "ARM"),
      color_manual = c(
        "150mg QD" = "#000000",
        "Placebo" = "#3498DB",
        "Combination" = "#E74C3C"
      ),
      shape_manual = c("N" = 1, "Y" = 2, "NA" = 0),
      plot_height = c(500, 200, 2000),
      facet_ncol = 2,
      trt_facet = FALSE,
      reg_line = FALSE,
      font_size = c(12, 8, 20),
      dot_size = c(1, 1, 12),
      reg_text_size = c(3, 3, 10)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_g_gh_spaghettiplot` *Spaghetti Plot*

Description

This teal module renders the UI and calls the function that creates a spaghetti plot.

Usage

```
tm_g_gh_spaghettiplot(
  label,
  dataname,
  param_var,
  param,
  param_var_label = "PARAM",
  idvar,
  xaxis_var,
  yaxis_var,
  xaxis_var_level = NULL,
  filter_var = yaxis_var,
  trt_group,
  trt_group_level = NULL,
  group_stats = "NONE",
  man_color = NULL,
  color_comb = NULL,
  xtick = ggplot2::waiver(),
  xlabel = xtick,
  rotate_xlab = FALSE,
  facet_ncol = 2,
  free_x = FALSE,
  plot_height = c(600, 200, 2000),
  plot_width = NULL,
  font_size = c(12, 8, 20),
  dot_size = c(2, 1, 12),
  hline_arb = numeric(0),
  hline_arb_color = "red",
  hline_arb_label = "Horizontal line",
  hline_vars = character(0),
  hline_vars_colors = "green",
  hline_vars_labels = hline_vars,
  pre_output = NULL,
  post_output = NULL
)
```

Arguments

`label`	menu item label of the module in the teal app.

dataname	analysis data passed to the data argument of init. E.g. ADaM structured laboratory data frame ADLB.
param_var	name of variable containing biomarker codes e.g. PARAMCD.
param	biomarker selected.
param_var_label	single name of variable in analysis data that includes parameter labels.
idvar	name of unique subject id variable.
xaxis_var	single name of variable in analysis data that is used as x-axis in the plot for the respective goshawk function.
yaxis_var	single name of variable in analysis data that is used as summary variable in the respective goshawk function.
xaxis_var_level	vector that can be used to define the factor level of xaxis_var. Only use it when xaxis_var is character or factor.
filter_var	data constraint variable.
trt_group	choices_selected object with available choices and pre-selected option for variable names representing treatment group e.g. ARM.
trt_group_level	vector that can be used to define factor level of trt_group.
group_stats	control group mean or median overlay.
man_color	string vector representing customized colors
color_comb	name or hex value for combined treatment color.
xtick	numeric vector to define the tick values of x-axis when x variable is numeric. Default value is waive().
xlabel	vector with same length of xtick to define the label of x-axis tick values. Default value is waive().
rotate_xlab	logical(1) value indicating whether to rotate x-axis labels
facet_ncol	numeric value indicating number of facets per row.
free_x	logical(1) should scales be "fixed" (FALSE) of "free" (TRUE) for x-axis in facet_wrap scales parameter.
plot_height	controls plot height.
plot_width	optional, controls plot width.
font_size	control font size for title, x-axis, y-axis and legend font.
dot_size	plot dot size.
hline_arb	numeric vector of at most 2 values identifying intercepts for arbitrary horizontal lines.
hline_arb_color	a character vector of at most length of hline_arb. naming the color for the arbitrary horizontal lines.
hline_arb_label	a character vector of at most length of hline_arb. naming the label for the arbitrary horizontal lines.

`hline_vars` a character vector to name the columns that will define additional horizontal lines.

`hline_vars_colors` a character vector naming the colors for the additional horizontal lines.

`hline_vars_labels` a character vector naming the labels for the additional horizontal lines that will appear in the legend.

`pre_output` (`shiny.tag`) optional, with text placed before the output to put the output into context. For example a title.

`post_output` (`shiny.tag`) optional, with text placed after the output to put the output into context. For example the `shiny::helpText()` elements are useful.

Value

`shiny` object

Author(s)

Wenyi Liu (luiw2) wenyi.liu@roche.com

Balazs Toth (tothb2) toth.balazs@gene.com

Examples

```
# Example using ADaM structure analysis dataset.
data <- teal_data()
data <- within(data, {
  library(dplyr)
  library(stringr)

  # use non-exported function from goshawk
  h_identify_loq_values <- getFromNamespace("h_identify_loq_values", "goshawk")

  # original ARM value = dose value
  arm_mapping <- list(
    "A: Drug X" = "150mg QD",
    "B: Placebo" = "Placebo",
    "C: Combination" = "Combination"
  )
  set.seed(1)
  ADSL <- rADSL
  ADLB <- rADLB
  var_labels <- lapply(ADLB, function(x) attributes(x)$label)
  ADLB <- ADLB %>%
    mutate(
      AVISITCD = case_when(
        AVISIT == "SCREENING" ~ "SCR",
        AVISIT == "BASELINE" ~ "BL",
        grepl("WEEK", AVISIT) ~ paste("W", str_extract(AVISIT, "(?<=(WEEK ))[0-9]+")),
        TRUE ~ as.character(NA)
```

```
      ),
      AVISITCDN = case_when(
        AVISITCD == "SCR" ~ -2,
        AVISITCD == "BL" ~ 0,
        grepl("W", AVISITCD) ~ as.numeric(gsub("[^0-9]*", "", AVISITCD)),
        TRUE ~ as.numeric(NA)
      ),
      AVISITCD = factor(AVISITCD) %>% reorder(AVISITCDN),
      TRTORD = case_when(
        ARMCD == "ARM C" ~ 1,
        ARMCD == "ARM B" ~ 2,
        ARMCD == "ARM A" ~ 3
      ),
      ARM = as.character(arm_mapping[match(ARM, names(arm_mapping))]),
      ARM = factor(ARM) %>% reorder(TRTORD),
      ACTARM = as.character(arm_mapping[match(ACTARM, names(arm_mapping))]),
      ACTARM = factor(ACTARM) %>% reorder(TRTORD),
      ANRLO = 30,
      ANRHI = 75
    ) %>%
    rowwise() %>%
    group_by(PARAMCD) %>%
    mutate(LBSTRESC = ifelse(USUBJID %in% sample(USUBJID, 1, replace = TRUE),
      paste("<", round(runif(1, min = 25, max = 30))), LBSTRESC
    )) %>%
    mutate(LBSTRESC = ifelse(USUBJID %in% sample(USUBJID, 1, replace = TRUE),
      paste(">", round(runif(1, min = 70, max = 75))), LBSTRESC
    )) %>%
    ungroup()
  attr(ADLB[["ARM"]], "label") <- var_labels[["ARM"]]
  attr(ADLB[["ACTARM"]], "label") <- var_labels[["ACTARM"]]
  attr(ADLB[["ANRLO"]], "label") <- "Analysis Normal Range Lower Limit"
  attr(ADLB[["ANRHI"]], "label") <- "Analysis Normal Range Upper Limit"

  # add LLOQ and ULOQ variables
  ALB_LOQS <- h_identify_loq_values(ADLB, "LOQFL")
  ADLB <- left_join(ADLB, ALB_LOQS, by = "PARAM")
})

datanames <- c("ADSL", "ADLB")
datanames(data) <- datanames
join_keys(data) <- default_cdisc_join_keys[datanames]

app <- init(
  data = data,
  modules = modules(
    tm_g_gh_spaghettiplot(
      label = "Spaghetti Plot",
      dataname = "ADLB",
      param_var = "PARAMCD",
      param = choices_selected(c("ALT", "CRP", "IGA"), "ALT"),
      idvar = "USUBJID",
      xaxis_var = choices_selected(c("Analysis Visit Code" = "AVISITCD"), "AVISITCD"),
```

```
      yaxis_var = choices_selected(c("AVAL", "CHG", "PCHG"), "AVAL"),
      filter_var = choices_selected(
        c("None" = "NONE", "Screening" = "BASE2", "Baseline" = "BASE"),
        "NONE"
      ),
      trt_group = choices_selected(c("ARM", "ACTARM"), "ARM"),
      color_comb = "#39ff14",
      man_color = c(
        "Combination" = "#000000",
        "Placebo" = "#fce300",
        "150mg QD" = "#5a2f5f"
      ),
      hline_arb = c(60, 50),
      hline_arb_color = c("grey", "red"),
      hline_arb_label = c("default A", "default B"),
      hline_vars = c("ANRHI", "ANRLO", "ULOQN", "LLOQN"),
      hline_vars_colors = c("pink", "brown", "purple", "black"),
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

Index