Package: teal.osprey (via r-universe)

August 31, 2024

Title R Package of Teal Module for TLG Functions in Osprey

Version 0.1.16

Date 2023-08-11

Description Community efforts to collect teal modules for TLGs defined in the osprey package. The teal modules add an encoding panel to interactively change the encoding within teal.

License Apache License 2.0 | file LICENSE

URL https://github.com/insightsengineering/teal.osprey/

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

Depends osprey (>= 0.1.16), R (>= 3.6), shiny, teal (>= 0.14.0)

Imports checkmate, dplyr, formatters (>= 0.3.1), ggplot2, lifecycle, logger (>= 0.2.0), shinyvalidate, teal.code (>= 0.4.0), teal.logger (>= 0.1.1), teal.reporter (>= 0.2.0), teal.transform (>= 0.4.0), teal.widgets (>= 0.4.0), tern (>= 0.7.10), tidyr

Suggests knitr, nestcolor (>= 0.1.0), rmarkdown, teal.data (>= 0.3.0), testthat (>= 2.0)

Config/Needs/website insightsengineering/nesttemplate

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

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

RemoteRef v0.1.16

RemoteSha 849b0111fba5bb4958e4474983f6cd6a08014f47

Contents

`label_aevar` *Automatically switch variable labels for standard* `AE` *variables in* `AE` *osprey functions* **[Stable]**

Description

Automatically switch variable labels for standard `AE` variables in `AE` osprey functions **[Stable]**

Usage

```
label_aevar(x)
```

Arguments

`x`	variable key

`plot_decorate_output` *Helper function to plot decorated output UI*

Description

[Stable]

Usage

```
plot_decorate_output(id)
```

Arguments

`id`	(`character`) id of this element

quick_filter *Utility function for quick filter* **[Stable]**

Description

Utility function for quick filter **[Stable]**

Usage

```
quick_filter(filter_opt, ANL)
```

Arguments

filter_opt	vector of string names of flag variable to filter (keep Y rows only)
ANL	input dataset

Value

a filtered dataframe

Author(s)

Carolyn Zhang (zhanc107) <carolyn.zhang@duke.edu>

srv_g_decorate *Helper server function to decorate plot output*

Description

[Stable]

This is used in tm_g_ae_oview and tm_g_events_term_id.

Usage

```
srv_g_decorate(
  id,
  plot_id = "out",
  plt = reactive(NULL),
  plot_height,
  plot_width
)
```

Arguments

id	(character) id of the module
plot_id	(character) id for plot output
plt	(reactive) a reactive object of graph object
plot_height	optional, (numeric) a vector of length three with c(value, min, max). Specifies the height of the main plot.
plot_width	optional, (numeric) a vector of length three with c(value, min, max). Specifies the width of the main plot and renders a slider on the plot to interactively adjust the plot width.

---

tm_g_ae_oview *Teal module for the* AE *overview*

Description

[Stable]

Display the AE overview plot as a shiny module

Usage

```
tm_g_ae_oview(
  label,
  dataname,
  arm_var,
  flag_var_anl,
  fontsize = c(5, 3, 7),
  plot_height = c(600L, 200L, 2000L),
  plot_width = NULL
)
```

Arguments

label	(character(1)) menu item label of the module in the teal app.
dataname	(character(1)) analysis data used in the teal module, needs to be available in the list passed to the data argument of teal::init().
arm_var	(choices_selected) object with all available choices and the pre-selected option for variable names that can be used as arm_var. See teal.transform::choices_selected() for details. Column arm_var in the dataname has to be a factor.
flag_var_anl	(teal.transform::choices_selected) choices_selected object with variables used to count adverse event sub-groups (e.g. Serious events, Related events, etc.)

fontsize (numeric(1) or numeric(3))
Defines initial possible range of font-size. fontsize is set for teal.widgets::optionalSliderInputVa
which controls font-size in the output plot.

plot_height (numeric(3))
vector to indicate default value, minimum and maximum values.

plot_width (numeric(3))
vector to indicate default value, minimum and maximum values.

Value

the teal::module() object.

Examples

```
library(nestcolor)

ADSL <- osprey::rADSL
ADAE <- osprey::rADAE

# Add additional dummy causality flags.
add_event_flags <- function(dat) {
  dat <- dat %>%
    dplyr::mutate(
      TMPFL_SER = AESER == "Y",
      TMPFL_REL = AEREL == "Y",
      TMPFL_GR5 = AETOXGR == "5",
      AEREL1 = (AEREL == "Y" & ACTARM == "A: Drug X"),
      AEREL2 = (AEREL == "Y" & ACTARM == "B: Placebo")
    )
  labels <- c(
    "Serious AE", "Related AE", "Grade 5 AE",
    "AE related to A: Drug X", "AE related to B: Placebo"
  )
  cols <- c("TMPFL_SER", "TMPFL_REL", "TMPFL_GR5", "AEREL1", "AEREL2")
  for (i in seq_along(labels)) {
    attr(dat[[cols[i]]], "label") <- labels[i]
  }
  dat
}
ADAE <- ADAE %>% add_event_flags()

app <- init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL, code = "ADSL <- osprey::rADSL"),
    cdisc_dataset("ADAE", ADAE,
      code =
        "ADAE <- osprey::rADAE
           add_event_flags <- function(dat) {
             dat <- dat %>%
               dplyr::mutate(
                 TMPFL_SER = AESER == 'Y',
                 TMPFL_REL = AEREL == 'Y',
```

```
                TMPFL_GR5 = AETOXGR == '5',
                AEREL1 = (AEREL == 'Y' & ACTARM == 'A: Drug X'),
                AEREL2 = (AEREL == 'Y' & ACTARM == 'B: Placebo')
              )
            labels <- c(
              'Serious AE',
              'Related AE',
              'Grade 5 AE',
              'AE related to A: Drug X',
              'AE related to B: Placebo'
            )
            cols <- c('TMPFL_SER', 'TMPFL_REL', 'TMPFL_GR5', 'AEREL1', 'AEREL2')
            for (i in seq_along(labels)) {
             attr(dat[[cols[i]]], 'label') <- labels[i]
            }
            dat
          }
          # Generating user-defined event flags.
          ADAE <- ADAE %>% add_event_flags()"
    ),
    check = TRUE
  ),
  modules = modules(
    tm_g_ae_oview(
      label = "AE Overview",
      dataname = "ADAE",
      arm_var = teal.transform::choices_selected(
        selected = "ACTARM",
        choices = c("ACTARM", "ACTARMCD")
      ),
      flag_var_anl = teal.transform::choices_selected(
        selected = "AEREL1",
        choices = teal.transform::variable_choices(
          ADAE,
          c("TMPFL_SER", "TMPFL_REL", "TMPFL_GR5", "AEREL1", "AEREL2")
        ),
      ),
      plot_height = c(600, 200, 2000)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_g_ae_sub *teal module for the* AE *by subgroups*

Description

[Stable]

Display the AE by subgroups plot as a teal module

Usage

```
tm_g_ae_sub(
  label,
  dataname,
  arm_var,
  group_var,
  plot_height = c(600L, 200L, 2000L),
  plot_width = NULL,
  fontsize = c(5, 3, 7)
)
```

Arguments

label	(character(1)) menu item label of the module in the teal app.
dataname	(character(1)) analysis data used in the teal module, needs to be available in the list passed to the data argument of teal::init().
arm_var	(choices_selected) object with all available choices and the pre-selected option for variable names that can be used as arm_var. See teal.transform::choices_selected() for details. Column arm_var in the dataname has to be a factor.
group_var	(choices_selected) subgroups variables. See teal.transform::choices_selected() for details.
plot_height	(numeric(3)) vector to indicate default value, minimum and maximum values.
plot_width	(numeric(3)) vector to indicate default value, minimum and maximum values.
fontsize	(numeric(1) or numeric(3)) Defines initial possible range of font-size. fontsize is set for teal.widgets::optionalSliderInputVa which controls font-size in the output plot.

Value

the teal::module() object.

Author(s)

Liming Li (Lil128) <liming.li@roche.com>

Molly He (hey59) <hey59@gene.com>

Examples

```
# Example using stream (ADaM) dataset
ADSL <- osprey::rADSL
ADAE <- osprey::rADAE

app <- init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL, code = "ADSL <- osprey::rADSL"),
    cdisc_dataset("ADAE", ADAE, code = "ADAE <- osprey::rADAE"),
    check = TRUE
  ),
  modules = modules(
    tm_g_ae_sub(
      label = "AE by Subgroup",
      dataname = "ADAE",
      arm_var = teal.transform::choices_selected(
        selected = "ACTARMCD",
        choices = c("ACTARM", "ACTARMCD")
      ),
      group_var = teal.transform::choices_selected(
        selected = c("SEX", "REGION1", "RACE"),
        choices = c("SEX", "REGION1", "RACE")
      ),
      plot_height = c(600, 200, 2000)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_g_butterfly *Butterfly plot Teal Module*

Description

[Stable]

Display butterfly plot as a shiny module

Usage

```
tm_g_butterfly(
  label,
  dataname,
  filter_var = NULL,
  right_var,
  left_var,
  category_var,
  color_by_var,
```

```
  count_by_var,
  facet_var = NULL,
 sort_by_var = teal.transform::choices_selected(selected = "count", choices = c("count",
   "alphabetical")),
  legend_on = TRUE,
  plot_height = c(600L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL
)
```

Arguments

label	(character(1)) menu item label of the module in the teal app.
dataname	(character(1)) analysis data used in the teal module, needs to be available in the list passed to the data argument of teal::init().
filter_var	(choices_selected) variable name of data filter, please see details regarding expected values, default is NULL. choices vector with filter_var choices, default is NULL
right_var	(choices_selected) dichotomization variable for right side
left_var	(choices_selected) dichotomization variable for left side
category_var	(choices_selected) category (y axis) variable
color_by_var	(choices_selected) variable defines color blocks within each bar
count_by_var	(choices_selected) variable defines how x axis is calculated
facet_var	(choices_selected) variable for row facets
sort_by_var	(choices_selected) argument for order of class and term elements in table, default here is "count"
legend_on	(boolean) value for whether legend is displayed
plot_height	(numeric(3)) vector to indicate default value, minimum and maximum values.
plot_width	(numeric(3)) vector to indicate default value, minimum and maximum 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

filter_var option is designed to work in conjunction with filtering function provided by teal (encoding panel on the right hand side of the shiny app). It can be used as quick access to predefined subsets of the domain datasets (not subject-level dataset) to be used for analysis, denoted by an

value of "Y". Each variable within the `filter_var_choices` is expected to contain values of either "Y" or "N". If multiple variables are selected as `filter_var`, only observations with "Y" value in each and every selected variables will be used for subsequent analysis. Flag variables (from `ADaM` datasets) can be used directly as filter.

Value

the `teal::module()` object.

Author(s)

Carolyn Zhang (zhanc107) <carolyn.zhang@duke.edu>

Chendi Liao (liaoc10) <chendi.liao@roche.com>

Examples

```
# Example using stream (ADaM) dataset
library(dplyr)
library(nestcolor)

set.seed(23)
ADSL <- osprey::rADSL
ADAE <- osprey::rADAE
ADSL <- mutate(ADSL, DOSE = paste(sample(1:3, n(), replace = TRUE), "UG"))
ADAE <- mutate(
  ADAE,
  flag1 = ifelse(AETOXGR == 1, 1, 0),
  flag2 = ifelse(AETOXGR == 2, 1, 0),
  flag3 = ifelse(AETOXGR == 3, 1, 0),
  flag1_filt = rep("Y", n())
)

app <- init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL,
      code = "ADSL <- osprey::rADSL
              set.seed(23)
             ADSL <- mutate(ADSL, DOSE = paste(sample(1:3, n(), replace = TRUE), 'UG'))"
    ),
    cdisc_dataset("ADAE", ADAE,
      code = "ADAE <- osprey::rADAE
              ADAE <- mutate(ADAE,
              flag1 = ifelse(AETOXGR == 1, 1, 0),
              flag2 = ifelse(AETOXGR == 2, 1, 0),
              flag3 = ifelse(AETOXGR == 3, 1, 0),
              flag1_filt = rep('Y', n()))"
    ),
    check = TRUE
  ),
  modules = modules(
    tm_g_butterfly(
      label = "Butterfly Plot",
```

```
      dataname = "ADAE",
      right_var = teal.transform::choices_selected(
        selected = "SEX",
        choices = c("SEX", "ARM", "RACE")
      ),
      left_var = teal.transform::choices_selected(
        selected = "RACE",
        choices = c("SEX", "ARM", "RACE")
      ),
      category_var = teal.transform::choices_selected(
        selected = "AEBODSYS",
        choices = c("AEDECOD", "AEBODSYS")
      ),
      color_by_var = teal.transform::choices_selected(
        selected = "AETOXGR",
        choices = c("AETOXGR", "None")
      ),
      count_by_var = teal.transform::choices_selected(
        selected = "# of patients",
        choices = c("# of patients", "# of AEs")
      ),
      facet_var = teal.transform::choices_selected(
        selected = NULL,
        choices = c("RACE", "SEX", "ARM")
      ),
      sort_by_var = teal.transform::choices_selected(
        selected = "count",
        choices = c("count", "alphabetical")
      ),
      legend_on = TRUE,
      plot_height = c(600, 200, 2000)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_g_events_term_id` *Events by Term Plot Teal Module*

Description

[Stable]

Display Events by Term plot as a shiny module

Usage

```
tm_g_events_term_id(
```

```
  label,
  dataname,
  term_var,
  arm_var,
  fontsize = c(5, 3, 7),
  plot_height = c(600L, 200L, 2000L),
  plot_width = NULL
)
```

Arguments

label	(character(1)) menu item label of the module in the teal app.
dataname	(character(1)) analysis data used in the teal module, needs to be available in the list passed to the data argument of teal::init().
term_var	choices_selected object with all available choices and pre-selected option names that can be used to specify the term for events
arm_var	(choices_selected) object with all available choices and the pre-selected option for variable names that can be used as arm_var. See teal.transform::choices_selected() for details. Column arm_var in the dataname has to be a factor.
fontsize	(numeric(1) or numeric(3)) Defines initial possible range of font-size. fontsize is set for teal.widgets::optionalSliderInputVa which controls font-size in the output plot.
plot_height	(numeric(3)) vector to indicate default value, minimum and maximum values.
plot_width	(numeric(3)) vector to indicate default value, minimum and maximum values.

Value

the teal::module() object.

Author(s)

Liming Li (lil128) <liming.li@roche.com>

Molly He (hey59) <hey59@gene.com>

Examples

```
library(nestcolor)

ADSL <- osprey::rADSL
ADAE <- osprey::rADAE

app <- init(
  data = cdisc_data(
```

```
    cdisc_dataset("ADSL", ADSL, code = "ADSL <- osprey::rADSL"),
    cdisc_dataset("ADAE", ADAE, code = "ADAE <- osprey::rADAE"),
    check = TRUE
  ),
  modules = modules(
    tm_g_events_term_id(
      label = "Common AE",
      dataname = "ADAE",
      term_var = teal.transform::choices_selected(
        selected = "AEDECOD",
        choices = c(
          "AEDECOD", "AETERM",
          "AEHLT", "AELLT", "AEBODSYS"
        )
      ),
      arm_var = teal.transform::choices_selected(
        selected = "ACTARMCD",
        choices = c("ACTARM", "ACTARMCD")
      ),
      plot_height = c(600, 200, 2000)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_g_heat_bygrade *Teal module for the heatmap by grade*

Description

[Stable]

Display the heatmap by grade as a shiny module

Usage

```
tm_g_heat_bygrade(
  label,
  sl_dataname,
  ex_dataname,
  ae_dataname,
  cm_dataname = NA,
  id_var,
  visit_var,
  ongo_var,
  anno_var,
  heat_var,
```

```
  conmed_var = NULL,
  fontsize = c(5, 3, 7),
  plot_height = c(600L, 200L, 2000L),
  plot_width = NULL
)
```

Arguments

label	(character(1)) menu item label of the module in the teal app.
sl_dataname	(character) subject level dataset name, needs to be available in the list passed to the data argument of init
ex_dataname	(character) exposures dataset name, needs to be available in the list passed to the data argument of init
ae_dataname	(character) adverse events dataset name, needs to be available in the list passed to the data argument of init
cm_dataname	(character) concomitant medications dataset name, needs to be available in the list passed to the data argument of init specify to NA if no concomitant medications data is available
id_var	(choices_seleced) unique subject ID variable
visit_var	(choices_seleced) analysis visit variable
ongo_var	(choices_seleced) study ongoing status variable, This variable is a derived logical variable. Usually it can be derived from EOSSTT.
anno_var	(choices_seleced) annotation variable
heat_var	(choices_seleced) heatmap variable
conmed_var	(choices_seleced) concomitant medications variable, specify to NA if no concomitant medications data is available
fontsize	(numeric(1) or numeric(3)) Defines initial possible range of font-size. fontsize is set for teal.widgets::optionalSliderInputVa which controls font-size in the output plot.
plot_height	(numeric(3)) vector to indicate default value, minimum and maximum values.
plot_width	(numeric(3)) vector to indicate default value, minimum and maximum values.

Value

the teal::module() object.

Examples

```
library(dplyr)
library(nestcolor)
```

```
ADSL <- osprey::rADSL %>% slice(1:30)
ADEX <- osprey::rADEX %>% filter(USUBJID %in% ADSL$USUBJID)
ADAE <- osprey::rADAE %>% filter(USUBJID %in% ADSL$USUBJID)
ADCM <- osprey::rADCM %>% filter(USUBJID %in% ADSL$USUBJID)

# This preprocess is only to force legacy standard on ADCM
ADCM <- ADCM %>%
  select(-starts_with("ATC")) %>%
  unique()

# function to derive AVISIT from ADEX
add_visit <- function(data_need_visit) {
  visit_dates <- ADEX %>%
    filter(PARAMCD == "DOSE") %>%
    distinct(USUBJID, AVISIT, ASTDTM) %>%
    group_by(USUBJID) %>%
    arrange(ASTDTM) %>%
    mutate(next_vis = lead(ASTDTM), is_last = ifelse(is.na(next_vis), TRUE, FALSE)) %>%
    rename(this_vis = ASTDTM)
  data_visit <- data_need_visit %>%
    select(USUBJID, ASTDTM) %>%
    left_join(visit_dates, by = "USUBJID") %>%
    filter(ASTDTM > this_vis & (ASTDTM < next_vis | is_last == TRUE)) %>%
    left_join(data_need_visit) %>%
    distinct()
  return(data_visit)
}
# derive AVISIT for ADAE and ADCM
ADAE <- add_visit(ADAE)
ADCM <- add_visit(ADCM)
# derive ongoing status variable for ADEX
ADEX <- ADEX %>%
  filter(PARCAT1 == "INDIVIDUAL") %>%
  mutate(ongo_status = (EOSSTT == "ONGOING"))

app <- init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL),
    cdisc_dataset("ADEX", ADEX),
    cdisc_dataset("ADAE", ADAE),
   cdisc_dataset("ADCM", ADCM, keys = c("STUDYID", "USUBJID", "ASTDTM", "CMSEQ", "CMDECOD")),
    code = "
    ADSL <- osprey::rADSL %>% slice(1:30)
    ADEX <- osprey::rADEX %>% filter(USUBJID %in% ADSL$USUBJID)
    ADAE <- osprey::rADAE %>% filter(USUBJID %in% ADSL$USUBJID)
    ADCM <- osprey::rADCM %>% filter(USUBJID %in% ADSL$USUBJID)
    ADCM <- ADCM %>% select(-starts_with(\"ATC\")) %>% unique()
    ADEX  <- ADEX %>%
      filter(PARCAT1 == 'INDIVIDUAL') %>%
      mutate(ongo_status = (EOSSTT == 'ONGOING'))
    add_visit <- function(data_need_visit) {
      visit_dates <- ADEX %>%
        filter(PARAMCD == 'DOSE') %>%
```

```
      distinct(USUBJID, AVISIT, ASTDTM) %>%
      group_by(USUBJID) %>%
      arrange(ASTDTM) %>%
     mutate(next_vis = lead(ASTDTM), is_last = ifelse(is.na(next_vis), TRUE, FALSE)) %>%
      rename(this_vis = ASTDTM)
    data_visit <- data_need_visit %>%
      select(USUBJID, ASTDTM) %>%
      left_join(visit_dates, by = 'USUBJID') %>%
      filter(ASTDTM > this_vis & (ASTDTM < next_vis | is_last == TRUE)) %>%
      left_join(data_need_visit) %>% distinct()
    return(data_visit)
  }
  ADAE <- add_visit(ADAE)
  ADCM <- add_visit(ADCM)
  ",
  check = TRUE
),
modules = modules(
  tm_g_heat_bygrade(
    label = "Heatmap by grade",
    sl_dataname = "ADSL",
    ex_dataname = "ADEX",
    ae_dataname = "ADAE",
    cm_dataname = "ADCM",
    id_var = teal.transform::choices_selected(
      selected = "USUBJID",
      choices = c("USUBJID", "SUBJID")
    ),
    visit_var = teal.transform::choices_selected(
      selected = "AVISIT",
      choices = c("AVISIT")
    ),
    ongo_var = teal.transform::choices_selected(
      selected = "ongo_status",
      choices = c("ongo_status")
    ),
    anno_var = teal.transform::choices_selected(
      selected = c("SEX", "COUNTRY"),
      choices = c("SEX", "COUNTRY", "USUBJID")
    ),
    heat_var = teal.transform::choices_selected(
      selected = "AETOXGR",
      choices = c("AETOXGR")
    ),
    conmed_var = teal.transform::choices_selected(
      selected = "CMDECOD",
      choices = c("CMDECOD")
    ),
    plot_height = c(600, 200, 2000)
  )
)
)
if (interactive()) {
```

```
  shinyApp(app$ui, app$server)
}
```

`tm_g_patient_profile` *Patient Profile plot teal module*

Description

[Stable]

Display patient profile plot as a shiny module

Usage

```
tm_g_patient_profile(
  label = "Patient Profile Plot",
  patient_id,
  sl_dataname,
  ex_dataname = NA,
  ae_dataname = NA,
  rs_dataname = NA,
  cm_dataname = NA,
  lb_dataname = NA,
  sl_start_date,
  ex_var = NULL,
  ae_var = NULL,
  ae_line_col_var = NULL,
  ae_line_col_opt = NULL,
  rs_var = NULL,
  cm_var = NULL,
  lb_var = NULL,
  x_limit = "-28, 365",
  plot_height = c(1200L, 400L, 5000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL
)
```

Arguments

`label`	(`character(1)`) menu item label of the module in the teal app.
`patient_id`	(`choices_seleced`) unique subject ID variable
`sl_dataname`	(`character`) subject level dataset name, needs to be available in the list passed to the `data` argument of `init`

`ex_dataname`, `ae_dataname`, `rs_dataname`, `cm_dataname`, `lb_dataname`
: (`character(1)`) names of exposure, adverse events, response, concomitant medications, and labs datasets, respectively; must be available in the list passed to the `data` argument of `init`
set to NA (default) to omit from analysis

`sl_start_date`
: (`choices_selected`) study start date variable, usually set to treatment start date or randomization date

`ex_var`
: (`choices_selected`) exposure variable to plot as each line
leave unspecified or set to `NULL` if exposure data is not available

`ae_var`
: (`choices_selected`) adverse event variable to plot as each line
leave unspecified or set to `NULL` if adverse events data is not available

`ae_line_col_var`
: (`choices_selected`) variable for coloring `AE` lines
leave unspecified or set to `NULL` if adverse events data is not available

`ae_line_col_opt`
: aesthetic values to map color values (named vector to map color values to each name). If not `NULL`, please make sure this contains all possible values for `ae_line_col_var` values.
leave unspecified or set to `NULL` if adverse events data is not available

`rs_var`
: (`choices_selected`) response variable to plot as each line
leave unspecified or set to `NULL` if response data is not available

`cm_var`
: (`choices_selected`) concomitant medication variable to plot as each line
leave unspecified or set to `NULL` if concomitant medications data is not available

`lb_var`
: (`choices_selected`) lab variable to plot as each line
leave unspecified or set to `NULL` if labs data is not available

`x_limit`
: a single `character` string with two numbers separated by a comma indicating the x-axis limit, default is `"-28, 365"`

`plot_height`
: (`numeric(3)`)
vector to indicate default value, minimum and maximum values.

`plot_width`
: (`numeric(3)`)
vector to indicate default value, minimum and maximum 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

As the patient profile module plots different domains in one plot, the study day (x-axis) is derived for consistency based the start date of user's choice in the app (for example, `ADSL.RANDDT` or `ADSL.TRTSDT`):

- In `ADAE`, `ADEX`, and `ADCM`, it would be study day based on `ASTDT` and/or `AENDT` in reference to the start date
- In `ADRS` and `ADLB`, it would be study day based on `ADT` in reference to the start date

Value

the teal::module() object.

Author(s)

Xuefeng Hou (houx14) <houx14@gene.com>

Tina Cho (chot) <tina.cho@roche.com>

Molly He (hey59) <hey59@gene.com>

Ting Qi (qit3) <qit3@gene.com>

Examples

```
library(nestcolor)

ADSL <- osprey::rADSL
ADAE <- osprey::rADAE %>%
  mutate(
    ASTDT = as.Date(ASTDTM),
    AENDT = as.Date(AENDTM)
  )
ADCM <- osprey::rADCM %>%
  mutate(
    ASTDT = as.Date(ASTDTM),
    AENDT = as.Date(AENDTM)
  )

# The step below is to pre-process ADCM to legacy standard
ADCM <- ADCM %>%
  select(-starts_with("ATC")) %>%
  unique()

ADRS <- osprey::rADRS %>%
  mutate(ADT = as.Date(ADTM))
ADEX <- osprey::rADEX %>%
  mutate(
    ASTDT = as.Date(ASTDTM),
    AENDT = as.Date(AENDTM)
  )
ADLB <- osprey::rADLB %>%
  mutate(
    ADT = as.Date(ADTM),
    LBSTRESN = as.numeric(LBSTRESC)
  )

app <- init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL, code = "ADSL <- osprey::rADSL"),
   cdisc_dataset("ADRS", ADRS, code = "ADRS <- osprey::rADRS %>% mutate(ADT = as.Date(ADTM))"),
    cdisc_dataset("ADAE", ADAE,
      code = "ADAE <- osprey::rADAE %>%
              mutate(ASTDT = as.Date(ASTDTM),
```

```
                  AENDT = as.Date(AENDTM))"
  ),
  cdisc_dataset("ADCM", ADCM,
    code = "ADCM <- osprey::rADCM %>%
            mutate(ASTDT = as.Date(ASTDTM),
                   AENDT = as.Date(AENDTM))
            ADCM <- ADCM %>% select(-starts_with(\"ATC\")) %>% unique()",
    keys = c("STUDYID", "USUBJID", "ASTDTM", "CMSEQ", "CMDECOD")
  ),
  cdisc_dataset("ADLB", ADLB,
    code = "ADLB <- osprey::rADLB %>%
            mutate(ADT = as.Date(ADTM),
                   LBSTRESN = as.numeric(LBSTRESC))"
  ),
  cdisc_dataset("ADEX", ADEX,
    code = "ADEX <- osprey::rADEX %>%
            mutate(ASTDT = as.Date(ASTDTM),
                   AENDT = as.Date(AENDTM))"
  ),
  check = FALSE # set FALSE here to keep run time of example short, should be set to TRUE
),
modules = modules(
  tm_g_patient_profile(
    label = "Patient Profile Plot",
    patient_id = teal.transform::choices_selected(
      choices = unique(ADSL$USUBJID),
      selected = unique(ADSL$USUBJID)[1]
    ),
    sl_dataname = "ADSL",
    ex_dataname = "ADEX",
    ae_dataname = "ADAE",
    rs_dataname = "ADRS",
    cm_dataname = "ADCM",
    lb_dataname = "ADLB",
    sl_start_date = teal.transform::choices_selected(
      selected = "TRTSDTM",
      choices = c("TRTSDTM", "RANDDT")
    ),
    ex_var = teal.transform::choices_selected(
      selected = "PARCAT2",
      choices = "PARCAT2"
    ),
    ae_var = teal.transform::choices_selected(
      selected = "AEDECOD",
      choices = c("AEDECOD", "AESOC")
    ),
    ae_line_col_var = teal.transform::choices_selected(
      selected = "AESER",
      choices = c("AESER", "AEREL")
    ),
    ae_line_col_opt = c("Y" = "red", "N" = "blue"),
    rs_var = teal.transform::choices_selected(
      selected = "PARAMCD",
```

```
        choices = "PARAMCD"
      ),
      cm_var = teal.transform::choices_selected(
        selected = "CMDECOD",
        choices = c("CMDECOD", "CMCAT")
      ),
      lb_var = teal.transform::choices_selected(
        selected = "LBTESTCD",
        choices = c("LBTESTCD", "LBCAT")
      ),
      x_limit = "-28, 750",
      plot_height = c(1200, 400, 5000)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_g_spiderplot` *Spider plot Teal Module*

Description

[Stable]

Display spider plot as a shiny module

Usage

```
tm_g_spiderplot(
  label,
  dataname,
  paramcd,
  x_var,
  y_var,
  marker_var,
  line_colorby_var,
  xfacet_var = NULL,
  yfacet_var = NULL,
  vref_line = NULL,
  href_line = NULL,
  anno_txt_var = TRUE,
  legend_on = FALSE,
  plot_height = c(600L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL
)
```

Arguments

label	(character(1)) menu item label of the module in the teal app.
dataname	(character(1)) analysis data used in the teal module, needs to be available in the list passed to the data argument of teal::init().
paramcd	(character(1) or choices_selected) variable value designating the studied parameter. See teal.transform::choices_selected() for details.
x_var	x-axis variables
y_var	y-axis variables
marker_var	variable dictates marker symbol
line_colorby_var	variable dictates line color
xfacet_var	variable for x facets
yfacet_var	variable for y facets
vref_line	vertical reference lines
href_line	horizontal reference lines
anno_txt_var	annotation text
legend_on	boolean value for whether legend is displayed
plot_height	(numeric(3)) vector to indicate default value, minimum and maximum values.
plot_width	(numeric(3)) vector to indicate default value, minimum and maximum 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.

Value

the teal::module() object.

Author(s)

Carolyn Zhang (zhanc107) <carolyn.zhang@duke.edu>

Chendi Liao (liaoc10) <chendi.liao@roche.com>

Examples

```
# Example using stream (ADaM) dataset
library(dplyr)
library(nestcolor)

ADSL <- osprey::rADSL
ADTR <- osprey::rADTR

app <- teal::init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL, code = "ADSL <- osprey::rADSL"),
    cdisc_dataset("ADTR", ADTR,
      code = "ADTR <- osprey::rADTR",
      keys = c("STUDYID", "USUBJID", "PARAMCD", "AVISIT")
    ),
    check = TRUE
  ),
  modules = modules(
    tm_g_spiderplot(
      label = "Spider plot",
      dataname = "ADTR",
      paramcd = teal.transform::choices_selected(
        choices = "SLDINV",
        selected = "SLDINV"
      ),
      x_var = teal.transform::choices_selected(
        choices = "ADY",
        selected = "ADY"
      ),
      y_var = teal.transform::choices_selected(
        choices = c("PCHG", "CHG", "AVAL"),
        selected = "PCHG"
      ),
      marker_var = teal.transform::choices_selected(
        choices = c("SEX", "RACE", "USUBJID"),
        selected = "SEX"
      ),
      line_colorby_var = teal.transform::choices_selected(
        choices = c("SEX", "USUBJID", "RACE"),
        selected = "SEX"
      ),
      xfacet_var = teal.transform::choices_selected(
        choices = c("SEX", "ARM"),
        selected = "SEX"
      ),
      yfacet_var = teal.transform::choices_selected(
        choices = c("SEX", "ARM"),
        selected = "ARM"
      ),
      vref_line = "10, 37",
      href_line = "-20, 0"
    )
```

```
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_g_swimlane` *Teal Module for* `Swimlane` *Plot*

Description

[Stable]

This is teal module that generates a `swimlane` plot (bar plot with markers) for `ADaM` data

Usage

```
tm_g_swimlane(
  label,
  dataname,
  bar_var,
  bar_color_var = NULL,
  sort_var = NULL,
  marker_pos_var = NULL,
  marker_shape_var = NULL,
  marker_shape_opt = NULL,
  marker_color_var = NULL,
  marker_color_opt = NULL,
  anno_txt_var = NULL,
  vref_line = NULL,
  plot_height = c(1200L, 400L, 5000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  x_label = "Time from First Treatment (Day)"
)
```

Arguments

`label`	(`character(1)`) menu item label of the module in the teal app.
`dataname`	analysis data used for plotting, needs to be available in the list passed to the `data` argument of `init`. If no markers are to be plotted in the module, `"ADSL"` should be the input. If markers are to be plotted, data name for the marker data should be the input
`bar_var`	(`choices_selected`) subject-level numeric variable from dataset to plot as the bar length

bar_color_var	(choices_selected) color by variable (subject-level)
sort_var	(choices_selected) sort by variable (subject-level)
marker_pos_var	(choices_selected) variable for marker position from marker data (Note: make sure that marker position has the same relative start day as bar length variable bar_var)
marker_shape_var	(choices_selected) marker shape variable from marker data
marker_shape_opt	aesthetic values to map shape values (named vector to map shape values to each name). If not NULL, please make sure this contains all possible values for marker_shape_var values, otherwise shape will be assigned by ggplot default
marker_color_var	marker color variable from marker data
marker_color_opt	aesthetic values to map color values (named vector to map color values to each name). If not NULL, please make sure this contains all possible values for marker_color_var values, otherwise color will be assigned by ggplot default
anno_txt_var	character vector with subject-level variable names that are selected as annotation
vref_line	vertical reference lines
plot_height	(numeric(3)) vector to indicate default value, minimum and maximum values.
plot_width	(numeric(3)) vector to indicate default value, minimum and maximum 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.
x_label	the label of the x axis

Value

the teal::module() object.

Author(s)

Ting Qi (qit3) <qit3@gene.com>

Examples

```
# Example using stream (ADaM) dataset
library(dplyr)
library(nestcolor)

ADSL <- osprey::rADSL %>%
  dplyr::mutate(TRTDURD = as.integer(TRTEDTM - TRTSDTM) + 1) %>%
```

```
  dplyr::filter(STRATA1 == "A" & ARMCD == "ARM A")
ADRS <- osprey::rADRS

ADRS <- ADRS %>%
  dplyr::filter(PARAMCD == "LSTASDI" & DCSREAS == "Death") %>%
  mutate(AVALC = DCSREAS, ADY = EOSDY) %>%
  base::rbind(ADRS %>% dplyr::filter(PARAMCD == "OVRINV" & AVALC != "NE")) %>%
  arrange(USUBJID)

app <- init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL, code = "ADSL <- osprey::rADSL %>%
      dplyr::mutate(TRTDURD = as.integer(TRTEDTM - TRTSDTM) + 1) %>%
      dplyr::filter(STRATA1 == 'A' & ARMCD == 'ARM A')"),
    cdisc_dataset("ADRS", ADRS,
      code = "ADRS <- rADRS
            ADRS <- ADRS %>% dplyr::filter(PARAMCD == 'LSTASDI' & DCSREAS == 'Death') %>%
                      mutate(AVALC = DCSREAS, ADY = EOSDY) %>%
              rbind(ADRS %>% dplyr::filter(PARAMCD == 'OVRINV' & AVALC != 'NE')) %>%
              arrange(USUBJID)"
    ),
    check = TRUE
  ),
  modules = modules(
    tm_g_swimlane(
      label = "Swimlane Plot",
      dataname = "ADRS",
      bar_var = teal.transform::choices_selected(
        selected = "TRTDURD",
        choices = c("TRTDURD", "EOSDY")
      ),
      bar_color_var = teal.transform::choices_selected(
        selected = "EOSSTT",
        choices = c("EOSSTT", "ARM", "ARMCD", "ACTARM", "ACTARMCD", "SEX")
      ),
      sort_var = teal.transform::choices_selected(
        selected = "ACTARMCD",
        choices = c("USUBJID", "SITEID", "ACTARMCD", "TRTDURD")
      ),
      marker_pos_var = teal.transform::choices_selected(
        selected = "ADY",
        choices = c("ADY")
      ),
      marker_shape_var = teal.transform::choices_selected(
        selected = "AVALC",
        c("AVALC", "AVISIT")
      ),
      marker_shape_opt = c("CR" = 16, "PR" = 17, "SD" = 18, "PD" = 15, "Death" = 8),
      marker_color_var = teal.transform::choices_selected(
        selected = "AVALC",
        choices = c("AVALC", "AVISIT")
      ),
      marker_color_opt = c(
```

```
          "CR" = "green", "PR" = "blue", "SD" = "goldenrod",
          "PD" = "red", "Death" = "black"
        ),
        vref_line = c(30, 60),
        anno_txt_var = teal.transform::choices_selected(
          selected = c("ACTARM", "SEX"),
          choices = c(
            "ARM", "ARMCD", "ACTARM", "ACTARMCD", "AGEGR1",
            "SEX", "RACE", "COUNTRY", "DCSREAS", "DCSREASP"
          )
        )
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

`tm_g_waterfall` *Teal Module for Waterfall Plot*

Description

[Stable]

This is teal module that generates a waterfall plot for `ADaM` data

Usage

```
tm_g_waterfall(
  label,
  dataname_tr = "ADTR",
  dataname_rs = "ADRS",
  bar_paramcd,
  bar_var,
  bar_color_var,
  bar_color_opt = NULL,
  sort_var,
  add_label_var_sl,
  add_label_paramcd_rs,
  anno_txt_var_sl,
  anno_txt_paramcd_rs,
  facet_var,
  ytick_at = 20,
  href_line = NULL,
  gap_point_val = NULL,
  show_value = TRUE,
  plot_height = c(1200L, 400L, 5000L),
```

```
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL
)
```

Arguments

label	(character(1)) menu item label of the module in the teal app.
dataname_tr	tumor burden analysis data used in teal module to plot as bar height, needs to be available in the list passed to the data argument of init
dataname_rs	response analysis data used in teal module to label response parameters, needs to be available in the list passed to the data argument of init
bar_paramcd	(choices_selected) parameter in tumor burden data that will be plotted as bar height
bar_var	(choices_selected) numeric variable from dataset to plot the bar height, e.g., PCHG
bar_color_var	(choices_selected) color by variable (subject level), None corresponds to NULL
bar_color_opt	aesthetic values to map color values (named vector to map color values to each name). If not NULL, please make sure this contains all possible values for bar_color_var values, otherwise color will be assigned by ggplot default, please note that NULL needs to be specified in this case
sort_var	(choices_selected) sort by variable (subject level), None corresponds to NULL
add_label_var_sl	(choices_selected) add label to bars (subject level), None corresponds to NULL
add_label_paramcd_rs	(choices_selected) add label to bars (response dataset), None corresponds to NULL. At least one of add_label_var_sl and add_label_paramcd_rs needs to be NULL
anno_txt_var_sl	(choices_selected) subject level variables to be displayed in the annotation table, default is NULL
anno_txt_paramcd_rs	(choices_selected) analysis dataset variables to be displayed in the annotation table, default is NULL
facet_var	(choices_selected) facet by variable (subject level), None corresponds to NULL
ytick_at	bar height axis interval, default is 20
href_line	numeric vector to plot horizontal reference lines, default is NULL
gap_point_val	singular numeric value for adding bar break when some bars are significantly higher than others, default is NULL
show_value	boolean of whether value of bar height is shown, default is TRUE
plot_height	(numeric(3)) vector to indicate default value, minimum and maximum values.

plot_width	(numeric(3)) vector to indicate default value, minimum and maximum 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.

Value

the teal::module() object.

Author(s)

Ting Qi (qit3) <qit3@gene.com>

houx14 <houx14@gene.com>

Examples

```
library(nestcolor)
ADSL <- osprey::rADSL
ADRS <- osprey::rADRS
ADTR <- osprey::rADTR

ADSL$SEX <- factor(ADSL$SEX, levels = unique(ADSL$SEX))

app <- teal::init(
  data = cdisc_data(
    cdisc_dataset("ADSL", ADSL,
      code = "ADSL <- rADSL
              ADSL$SEX <- factor(ADSL$SEX, levels = unique(ADSL$SEX))"
    ),
    cdisc_dataset("ADRS", ADRS, code = "ADRS <- rADRS"),
    cdisc_dataset("ADTR", ADTR,
      code = " ADTR <- rADTR",
      c("STUDYID", "USUBJID", "PARAMCD", "AVISIT")
    ),
    check = TRUE
  ),
  modules = modules(
    tm_g_waterfall(
      label = "Waterfall",
      dataname_tr = "ADTR",
      dataname_rs = "ADRS",
      bar_paramcd = teal.transform::choices_selected(c("SLDINV"), "SLDINV"),
      bar_var = teal.transform::choices_selected(c("PCHG", "AVAL"), "PCHG"),
      bar_color_var = teal.transform::choices_selected(c("ARMCD", "SEX"), "ARMCD"),
      bar_color_opt = NULL,
      sort_var = teal.transform::choices_selected(c("ARMCD", "SEX"), NULL),
      add_label_var_sl = teal.transform::choices_selected(c("SEX", "EOSDY"), NULL),
    add_label_paramcd_rs = teal.transform::choices_selected(c("BESRSPI", "OBJRSPI"), NULL),
```

```
    anno_txt_var_sl = teal.transform::choices_selected(c("SEX", "ARMCD", "BMK1", "BMK2"), NULL),
    anno_txt_paramcd_rs = teal.transform::choices_selected(c("BESRSPI", "OBJRSPI"), NULL),
    facet_var = teal.transform::choices_selected(c("SEX", "ARMCD", "STRATA1", "STRATA2"), NULL),
      href_line = "-30, 20"
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`ui_g_decorate` *Helper UI function to decorate plot output UI*

Description

[Stable]

This is used in `tm_g_ae_oview` and `tm_g_events_term_id`.

Usage

```
ui_g_decorate(
  id,
  titles = "Titles",
  footnotes = "footnotes",
  fontsize = c(5, 4, 11)
)
```

Arguments

`id`	(`character`) id of this module. set to `NULL` if you want to make it identical to the module who called it.
`titles`	(`character`) default titles
`footnotes`	(`character`) default footnotes
`fontsize`	(`numeric(1)` or `numeric(3)`) Defines initial possible range of font-size. `fontsize` is set for `teal.widgets::optionalSliderInputVa` which controls font-size in the output plot.

Index