# Package: teal.goshawk (via r-universe)

October 11, 2026

**Type** Package

**Title** Longitudinal Visualization `teal` Modules

**Version** 0.6.1

**Date** 2026-09-10

**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 (>= 0.1.19), R (>= 3.6), shiny (>= 1.8.1), teal (>= 1.1.0), teal.picks (>= 0.3.0), teal.transform (>= 0.7.0)

**Imports** bslib (>= 0.8.0), 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 (>= 2.1.0), shinyvalidate (>= 0.1.3), stats, teal.code (>= 0.7.0), teal.logger (>= 0.4.0), teal.reporter (>= 0.6.0), teal.widgets (>= 0.5.0)

**Suggests** jsonlite (>= 2.0.0), knitr (>= 1.42), logger (>= 0.4.0), nestcolor (>= 0.1.0), rmarkdown (>= 2.23), rvest (>= 1.0.0), shinytest2 (>= 0.4.1), stringr (>= 1.4.1), teal.data (>= 0.8.0), tern (>= 0.9.7), testthat (>= 3.2.3), utils, withr (>= 3.0.0)

**VignetteBuilder** knitr, rmarkdown

**Remotes** insightsengineering/goshawk@*release

**Config/Needs/verdepcheck** insightsengineering/goshawk, rstudio/shiny, insightsengineering/teal, insightsengineering/teal.transform, rstudio/bslib, mllg/checkmate, daattali/colourpicker, tidyverse/dplyr, rstudio/DT, tidyverse/ggplot2, r-lib/rlang, daattali/shinyjs, rstudio/shinyvalidate, insightsengineering/teal.code, insightsengineering/teal.logger, insightsengineering/teal.reporter, insightsengineering/teal.widgets, yihui/knitr, insightsengineering/nestcolor, rstudio/rmarkdown, tidyverse/rvest, rstudio/shinytest2, tidyverse/stringr, insightsengineering/teal.data, insightsengineering/tern, r-lib/testthat

**Config/Needs/website** insightsengineering/nesttemplate

**Config/roxygen2/version** 8.1.0

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**Roxygen** list(markdown = TRUE)

**Config/pak/sysreqs** libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libnode-dev zlib1g-dev

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

**Date/Publication** 2026-09-10 19:37:46 UTC

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

**RemoteRef** v0.6.1

**RemoteSha** 1d08610cad3870a4516ef258116a4cfd754a8d3c

# Contents

maptrt . . . . . 3
tm_g_gh_boxplot . . . . . 4
tm_g_gh_correlationplot . . . . . 8
tm_g_gh_density_distribution_plot . . . . . 15
tm_g_gh_lineplot . . . . . 19
tm_g_gh_spaghettiplot . . . . . 24

**Index** **30**

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.

## Value

Data frame with the unique treatment codes. The function prints the left hand side of the mapping or ordering code to the console.

## Examples

```
ADSL <- teal.data::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 = "ADLB",
  param_var = lifecycle::deprecated(),
  param = teal.picks::picks(teal.picks::variables("PARAMCD", "PARAMCD"),
   teal.picks::values(selected = "ALT", multiple = FALSE), check_dataset = FALSE),
  yaxis_var = teal.picks::variables(c("AVAL", "CHG"), "AVAL"),
  xaxis_var = teal.picks::variables("AVISITCD", "AVISITCD"),
  facet_var = teal.picks::variables(dplyr::starts_with("ARM"), selected = "ARM"),
  trt_group = teal.picks::variables(selected = "ARM"),
  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,
  transformators = list(),
  decorators = list()
)
```

### Arguments

| | |
|---|---|
| `label` | (`character(1)`) menu item label of the module in the teal app. |
| `dataname` | (`character(1)`) analysis data passed to the data argument of `teal::init()`. E.g. `ADaM` structured laboratory data frame `ADLB`. |

param_var **[Deprecated]** (character(1)) name of variable containing biomarker codes e.g. PARAMCD.

param (teal.picks::picks() or teal.transform::choices_selected()) biomarker selected.

yaxis_var (teal.picks::variables() or legacy teal.transform::choices_selected()) name of variable containing biomarker results displayed on y-axis e.g. AVAL.

xaxis_var (teal.picks::variables() or legacy teal.transform::choices_selected()) name of variable containing biomarker results displayed on x-axis e.g. BASE.

facet_var (variables or choices_selected) object with available choices and pre-selected option for variable names representing facet variable e.g. AVISITCD.

trt_group (teal.picks::variables() or legacy teal.transform::choices_selected()) object with available choices and pre-selected option for variable names representing treatment group e.g. ARM.

color_manual (named character, optional) vector of colors applied to treatment values.

shape_manual (named numeric, optional) vector of symbols applied to LOQ values.

facet_ncol (integer(1)) numeric value indicating number of facets per row.

loq_legend (logical(1)) loq legend toggle.

rotate_xlab (logical(1)) 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 (character) a character vector of at most length of hline_arb. naming the color for the arbitrary horizontal lines.

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

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

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

hline_vars_labels (character) a character vector naming the labels for the additional horizontal lines that will appear in the plot.

plot_height (numeric(3)) controls plot height.

plot_width (numeric(3), optional) controls plot width.

font_size (numeric(3)) font size control for title, x-axis label, y-axis label and legend.

dot_size (numeric(3)) plot dot size.

alpha (numeric(3)) 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.

transformators (list of teal_transform_module) that will be applied to transform module's data input. To learn more check vignette("transform-input-data", package = "teal").

decorators **[Experimental]** (named list of lists of teal_transform_module) optional, decorator for tables or plots included in the module output reported. The decorators are applied to the respective output objects.
See section "Decorating Module" below for more details.

## Value

A teal::module() object that can be used in a teal::init() call.

## Decorating Module

This module generates the following objects, which can be modified in place using decorators:

- plot (ggplot)

A Decorator is applied to the specific output using a named list of teal_transform_module objects. The name of this list corresponds to the name of the output to which the decorator is applied. See code snippet below:

```
tm_g_gh_boxplot(
   ..., # arguments for module
   decorators = list(
     plot = teal_transform_module(...) # applied only to `plot` output
   )
)
```

For additional details and examples of decorators, refer to the vignette vignette("decorate-module-output", package = "teal.goshawk").

To learn more please refer to the vignette vignette("transform-module-output", package = "teal") or the teal::teal_transform_module() documentation.

## Reporting

This module returns an object of class teal_module, that contains a server function. Since the server function returns a teal_report object, this makes this module reportable, which means that the reporting functionality will be turned on automatically by the teal framework.

For more information on reporting in teal, see the vignettes:

- vignette("reportable-shiny-application", package = "teal.reporter")
- vignette("adding-support-for-reporting-to-custom-modules", package = "teal")

## Author(s)

Jeff Tomlinson

Balazs Toth

## 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"
  )
  ADSL <- teal.data::rADSL
  ADLB <- teal.data::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")
})

join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_g_gh_boxplot(
      label = "Box Plot",
      dataname = "ADLB",
      param = picks(
        variables("PARAMCD", "PARAMCD"),
        values(selected = "ALT", multiple = FALSE),
        check_dataset = FALSE
      ),
      yaxis_var = variables(c("AVAL", "BASE", "CHG"), "AVAL"),
      xaxis_var = variables(c("ACTARM", "ARM", "AVISITCD", "STUDYID"), "ARM"),
      facet_var = variables(c("ACTARM", "ARM", "AVISITCD", "SEX"), "AVISITCD"),
      trt_group = variables(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 = "ADLB",
  param_var = lifecycle::deprecated(),
  xaxis_param = teal.picks::picks(teal.picks::variables("PARAMCD", "PARAMCD"),
   teal.picks::values(selected = "ALT", multiple = FALSE), check_dataset = FALSE),
  xaxis_var = teal.picks::variables(c("AVAL", "BASE", "CHG", "PCHG"), "BASE"),
  yaxis_param = teal.picks::picks(teal.picks::variables("PARAMCD", "PARAMCD"),
   teal.picks::values(selected = "CRP", multiple = FALSE), check_dataset = FALSE),
  yaxis_var = teal.picks::variables(c("AVAL", "BASE", "CHG", "PCHG"), "AVAL"),
  trt_group = teal.picks::variables(dplyr::starts_with("ARM"), selected = "ARM"),
  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,
```

```
  transformators = list(),
  decorators = list()
)
```

## Arguments

| | |
|---|---|
| label | (character(1)) menu item label of the module in the teal app. |
| dataname | (character(1)) analysis data passed to the data argument of teal::init(). E.g. ADaM structured laboratory data frame ADLB. |
| param_var | **[Deprecated]** (character(1)) name of variable containing biomarker codes e.g. PARAMCD. |
| xaxis_param | (teal.picks::picks() or legacy teal.transform::choices_selected()) biomarker selected for x-axis. |
| xaxis_var | (teal.picks::variables() or legacy teal.transform::choices_selected()) name of variable containing biomarker results displayed on x-axis e.g. BASE. |
| yaxis_param | (teal.picks::picks() or legacy teal.transform::choices_selected()) biomarker selected for y-axis. |
| yaxis_var | (teal.picks::variables() or legacy teal.transform::choices_selected()) name of variable containing biomarker results displayed on y-axis e.g. AVAL. |
| trt_group | (teal.picks::variables() or legacy teal.transform::choices_selected()) object with available choices and pre-selected option for variable names representing treatment group e.g. ARM. |
| color_manual | (named character, optional) vector of colors applied to treatment values. |
| shape_manual | (named numeric, optional) vector of symbols applied to LOQ values. |
| facet_ncol | (integer(1)) numeric value indicating number of facets per row. |
| visit_facet | (logical(1)) visit facet toggle. |
| trt_facet | (logical(1)) facet by treatment group trt_group. |
| reg_line | (logical(1)) include regression line and annotations for slope and coefficient in visualization. Use with facet TRUE. |
| loq_legend | (logical(1)) loq legend toggle. |
| rotate_xlab | (logical(1)) 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 | (character) a character vector of at most length of hline_arb. naming the color for the arbitrary horizontal lines. |
| hline_arb_label | (character) a character vector of at most length of hline_arb. naming the label for the arbitrary horizontal lines. |
| hline_vars | (character) a character vector to name the columns that will define additional horizontal lines. |
| hline_vars_colors | (character) a character vector naming the colors for the additional horizontal lines. |

`hline_vars_labels`
: (`character`) a character vector naming the labels for the additional horizontal lines that will appear in the plot.

`vline_arb`
: (`numeric`) vector of at most 2 values identifying intercepts for arbitrary vertical lines.

`vline_arb_color`
: (`character`) a character vector of at most length of `vline_arb`. naming the color for the arbitrary vertical lines.

`vline_arb_label`
: (`character`) a character vector of at most length of `vline_arb`. naming the label for the arbitrary vertical lines.

`vline_vars`
: (`character`) a character vector to name the columns that will define additional vertical lines.

`vline_vars_colors`
: (`character`) a character vector naming the colors for the additional vertical lines.

`vline_vars_labels`
: (`character`) a character vector naming the labels for the additional vertical lines that will appear in the plot.

`plot_height`
: (`numeric(3)`) controls plot height.

`plot_width`
: (`numeric(3)`, optional) controls plot width.

`font_size`
: (`numeric(3)`) font size control for title, `x-axis` label, `y-axis` label and legend.

`dot_size`
: (`numeric(3)`) plot dot size.

`reg_text_size`
: (`numeric(3)`) 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.

`transformators`
: (`list` of `teal_transform_module`) that will be applied to transform module's data input. To learn more check `vignette("transform-input-data", package = "teal")`.

`decorators`
: **[Experimental]** (named `list` of lists of `teal_transform_module`) optional, decorator for tables or plots included in the module output reported. The decorators are applied to the respective output objects.
See section "Decorating Module" below for more details.

## Value

A `teal::module()` object that can be used in a `teal::init()` call.

## Decorating Module

This module generates the following objects, which can be modified in place using decorators:

- `plot` (`ggplot`)

A Decorator is applied to the specific output using a named list of `teal_transform_module` objects. The name of this list corresponds to the name of the output to which the decorator is applied. See code snippet below:

```
tm_g_gh_correlationplot(
   ..., # arguments for module
   decorators = list(
     plot = teal_transform_module(...) # applied only to `plot` output
   )
)
```

For additional details and examples of decorators, refer to the vignette `vignette("decorate-module-output", package = "teal.goshawk")`.

To learn more please refer to the vignette `vignette("transform-module-output", package = "teal")` or the `teal::teal_transform_module()` documentation.

## Reporting

This module returns an object of class `teal_module`, that contains a `server` function. Since the server function returns a `teal_report` object, this makes this module reportable, which means that the reporting functionality will be turned on automatically by the `teal` framework.

For more information on reporting in `teal`, see the vignettes:

- `vignette("reportable-shiny-application", package = "teal.reporter")`
- `vignette("adding-support-for-reporting-to-custom-modules", package = "teal")`

## Author(s)

Nick Paszty

Balazs Toth

## 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)

ADSL <- teal.data::rADSL
ADLB <- teal.data::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")
})

join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_g_gh_correlationplot(
      label = "Correlation Plot",
      dataname = "ADLB",
      xaxis_param = picks(
        variables("PARAMCD", "PARAMCD"),
        values(selected = "ALT", multiple = FALSE),
        check_dataset = FALSE
      ),
      yaxis_param = picks(
        variables("PARAMCD", "PARAMCD"),
        values(selected = "CRP", multiple = FALSE),
        check_dataset = FALSE
      ),
      xaxis_var = variables(c("AVAL", "BASE", "CHG", "PCHG"), "BASE"),
      yaxis_var = variables(c("AVAL", "BASE", "CHG", "PCHG"), "AVAL"),
      trt_group = variables(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 = "ADLB",
  param_var = lifecycle::deprecated(),
  param = teal.picks::picks(teal.picks::variables("PARAMCD", "PARAMCD"),
   teal.picks::values(selected = "ALT", multiple = FALSE), check_dataset = FALSE),
  xaxis_var = teal.picks::variables(c("AVAL", "BASE", "CHG", "PCHG"), "AVAL"),
  trt_group = teal.picks::variables(dplyr::starts_with("ARM"), selected = "ARM"),
  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,
  transformators = list(),
  decorators = list()
)
```

## Arguments

| | |
|---|---|
| label | (character(1)) menu item label of the module in the teal app. |
| dataname | (character(1)) analysis data passed to the data argument of teal::init(). E.g. ADaM structured laboratory data frame ADLB. |
| param_var | **[Deprecated]** (character(1)) name of variable containing biomarker codes e.g. PARAMCD. |
| param | (teal.picks::picks() or teal.transform::choices_selected()) biomarker selected. |
| xaxis_var | (teal.picks::variables() or legacy teal.transform::choices_selected()) name of variable containing biomarker results displayed on x-axis e.g. BASE. |
| trt_group | (teal.picks::variables() or legacy teal.transform::choices_selected()) object with available choices and pre-selected option for variable names representing treatment group e.g. ARM. |
| color_manual | (named character, optional) vector of colors applied to treatment values. |
| color_comb | name or hex value for combined treatment color. |
| plot_height | (numeric(3)) controls plot height. |
| plot_width | (numeric(3), optional) controls plot width. |
| font_size | (numeric(3)) 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 | (character) a character vector of at most length of hline_arb. naming the color for the arbitrary horizontal lines. |
| hline_arb_label | (character) a character vector of at most length of hline_arb. naming the label for the arbitrary horizontal lines. |
| facet_ncol | (integer(1)) numeric value indicating number of facets per row. |
| comb_line | display combined treatment line toggle. |
| rotate_xlab | (logical(1)) 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.

transformators (list of teal_transform_module) that will be applied to transform module's data input. To learn more check vignette("transform-input-data", package = "teal").

decorators **[Experimental]** (named list of lists of teal_transform_module) optional, decorator for tables or plots included in the module output reported. The decorators are applied to the respective output objects.
See section "Decorating Module" below for more details.

## Value

A teal::module() object that can be used in a teal::init() call.

## Decorating Module

This module generates the following objects, which can be modified in place using decorators:

- plot (ggplot)

A Decorator is applied to the specific output using a named list of teal_transform_module objects. The name of this list corresponds to the name of the output to which the decorator is applied. See code snippet below:

```
tm_g_gh_density_distribution_plot(
   ..., # arguments for module
   decorators = list(
     plot = teal_transform_module(...) # applied only to `plot` output
   )
)
```

For additional details and examples of decorators, refer to the vignette vignette("decorate-module-output", package = "teal.goshawk").

To learn more please refer to the vignette vignette("transform-module-output", package = "teal") or the teal::teal_transform_module() documentation.

## Reporting

This module returns an object of class teal_module, that contains a server function. Since the server function returns a teal_report object, this makes this module reportable, which means that the reporting functionality will be turned on automatically by the teal framework.

For more information on reporting in teal, see the vignettes:

- vignette("reportable-shiny-application", package = "teal.reporter")
- vignette("adding-support-for-reporting-to-custom-modules", package = "teal")

## Author(s)

Nick Paszty

Balazs Toth

## 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 <- teal.data::rADSL
  ADLB <- teal.data::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"]]
})

join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_g_gh_density_distribution_plot(
```

```
        label = "Density Distribution Plot",
        dataname = "ADLB",
        param = picks(
          variables("PARAMCD", "PARAMCD"),
          values(selected = "ALT", multiple = FALSE),
          check_dataset = FALSE
        ),
        xaxis_var = variables(c("AVAL", "BASE", "CHG", "PCHG"), "AVAL"),
        trt_group = variables(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 = "ADLB",
  param_var = lifecycle::deprecated(),
  param = teal.picks::picks(teal.picks::variables("PARAMCD", "PARAMCD"),
   teal.picks::values(selected = "ALT", multiple = FALSE), check_dataset = FALSE),
  param_var_label = "PARAM",
  xaxis_var = teal.picks::variables(dplyr::starts_with("AVISIT"), "AVISITCD"),
  yaxis_var = teal.picks::variables(c("AVAL", "CHG", "PCHG"), "AVAL"),
  xvar_level = NULL,
```

```
  filter_var = lifecycle::deprecated(),
  filter_var_choices = lifecycle::deprecated(),
  trt_group = teal.picks::variables(selected = "ARM"),
  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,
  transformators = list(),
  decorators = list()
)
```

## Arguments

- `label` — (`character(1)`) Label shown in the navigation item for the module or module group. For `modules()` defaults to `"root"`. See `Details`.
- `dataname` — (`character(1)`) analysis data passed to the data argument of `teal::init()`. E.g. `ADaM` structured laboratory data frame `ADLB`.
- `param_var` — **[Deprecated]** (`character(1)`) name of variable containing biomarker codes e.g. `PARAMCD`.
- `param` — (`teal.picks::picks()` or `teal.transform::choices_selected()`) biomarker selected.
- `param_var_label` — (`character(1)`) single name of variable in analysis data that includes parameter labels.
- `xaxis_var` — (`teal.picks::variables()` or legacy `teal.transform::choices_selected()`) name of variable containing biomarker results displayed on x-axis e.g. `BASE`.
- `yaxis_var` — (`teal.picks::variables()` or legacy `teal.transform::choices_selected()`) name of variable containing biomarker results displayed on y-axis e.g. `AVAL`.
- `xvar_level` — (`character()`) vector that can be used to define the factor level of `xvar`. Only use it when `xaxis_var` is of type character or factor.

filter_var **[Deprecated]** data constraint variable.

filter_var_choices
**[Deprecated]** data constraint variable choices.

trt_group (teal.picks::variables() or legacy teal.transform::choices_selected()) object with available choices and pre-selected option for variable names representing treatment group e.g. ARM.

trt_group_level
(named character()) vector that can be used to define factor level of trt_group.

shape_choices (character(), teal.picks::variables() or legacy teal.transform::choices_selected()) vector or teal.picks::variables() object with names of ADSL variables which can be used to change shape

stat (character(1)) string of statistics

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

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

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

color_manual (named character, optional) vector of colors applied to treatment values.

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

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

rotate_xlab (logical(1)) 45 degree rotation of x-axis values.

plot_height (numeric(3)) controls plot height.

plot_width (numeric(3), optional) controls plot width.

plot_font_size (numeric(3)) control font size for title, x-axis, y-axis and legend font.

dodge (numeric(3)) 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
(numeric(1)) minimum count of observations (as listed in the output table) to plot nodes on the graph

table_font_size
(numeric(3)) controls the font size of values in the table.

dot_size (numeric(3)) plot dot size.

plot_relative_height_value
(numeric(1)) numeric value between 500 and 5000 for controlling the starting value of the relative plot height slider

transformators (list of teal_transform_module) that will be applied to transform module's data input. To learn more check vignette("transform-input-data", package = "teal").

decorators **[Experimental]** (named list of lists of teal_transform_module) optional, decorator for tables or plots included in the module output reported. The decorators are applied to the respective output objects.

See section "Decorating Module" below for more details.

## Value

A teal::module() object that can be used in a teal::init() call.

## Decorating Module

This module generates the following objects, which can be modified in place using decorators:

- plot (ggplot)

A Decorator is applied to the specific output using a named list of teal_transform_module objects. The name of this list corresponds to the name of the output to which the decorator is applied. See code snippet below:

```
tm_g_gh_lineplot(
   ..., # arguments for module
   decorators = list(
     plot = teal_transform_module(...) # applied only to `plot` output
   )
)
```

For additional details and examples of decorators, refer to the vignette vignette("decorate-module-output", package = "teal.goshawk").

To learn more please refer to the vignette vignette("transform-module-output", package = "teal") or the teal::teal_transform_module() documentation.

## Reporting

This module returns an object of class teal_module, that contains a server function. Since the server function returns a teal_report object, this makes this module reportable, which means that the reporting functionality will be turned on automatically by the teal framework.

For more information on reporting in teal, see the vignettes:

- vignette("reportable-shiny-application", package = "teal.reporter")
- vignette("adding-support-for-reporting-to-custom-modules", package = "teal")

## Author(s)

Wenyi Liu

Balazs Toth

## 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 <- teal.data::rADSL
  ADLB <- teal.data::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"]]
})

join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_g_gh_lineplot(
```

```
        label = "Line Plot",
        dataname = "ADLB",
        param = picks(
          variables("PARAMCD", "PARAMCD"),
          values(selected = "ALT", multiple = FALSE),
          check_dataset = FALSE
        ),
        shape_choices = c("SEX", "RACE"),
        xaxis_var = variables("AVISITCD", "AVISITCD"),
        yaxis_var = variables(c("AVAL", "BASE", "CHG", "PCHG"), "AVAL"),
        trt_group = variables(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_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 = "ADLB",
  param_var = lifecycle::deprecated(),
  param = teal.picks::picks(teal.picks::variables("PARAMCD", "PARAMCD"),
   teal.picks::values(selected = "ALT", multiple = FALSE), check_dataset = FALSE),
  param_var_label = "PARAM",
  idvar = "USUBJID",
  xaxis_var = teal.picks::variables(c("AVISITCD", "AVISIT"), "AVISITCD"),
  yaxis_var = teal.picks::variables(c("AVAL", "CHG", "PCHG"), "AVAL"),
  xaxis_var_level = NULL,
  filter_var = lifecycle::deprecated(),
  trt_group = teal.picks::variables(dplyr::starts_with("ARM"), selected = "ARM"),
  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,
  alpha = c(0.8, 0, 1),
  pre_output = NULL,
  post_output = NULL,
  transformators = list(),
  decorators = list()
)
```

## 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 | **[Deprecated]** (`character(1)`) name of variable containing biomarker codes e.g. `PARAMCD`. |
| param | (`teal.picks::picks()` or `teal.transform::choices_selected()`) biomarker selected. |
| param_var_label | (`character(1)`) single name of variable in analysis data that includes parameter labels. |
| idvar | name of unique subject id variable. |
| xaxis_var | (`teal.picks::variables()` or legacy `teal.transform::choices_selected()`) name of variable containing biomarker results displayed on x-axis e.g. `BASE`. |
| yaxis_var | (`teal.picks::variables()` or legacy `teal.transform::choices_selected()`) name of variable containing biomarker results displayed on y-axis e.g. `AVAL`. |
| 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 | **[Deprecated]** data constraint variable. |
| trt_group | (`teal.picks::variables()` or legacy `teal.transform::choices_selected()`) object with available choices and pre-selected option for variable names representing treatment group e.g. `ARM`. |

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

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

`rotate_xlab`
: (`logical(1)`) 45 degree rotation of `x-axis` values.

`facet_ncol`
: (`integer(1)`) 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`
: (`numeric(3)`) controls plot height.

`plot_width`
: (`numeric(3)`, optional) controls plot width.

`font_size`
: (`numeric(3)`) font size control for title, `x-axis` label, `y-axis` label and legend.

`dot_size`
: (`numeric(3)`) plot dot size.

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

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

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

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

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

`hline_vars_labels`
: (`character`) a character vector naming the labels for the additional horizontal lines that will appear in the plot.

`alpha`
: (`numeric(3)`) 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.

`transformators`
: (`list` of `teal_transform_module`) that will be applied to transform module's data input. To learn more check `vignette("transform-input-data", package = "teal")`.

`decorators`
: **[Experimental]** (named `list` of lists of `teal_transform_module`) optional, decorator for tables or plots included in the module output reported. The decorators are applied to the respective output objects.
See section "Decorating Module" below for more details.

### Value

A `teal::module()` object that can be used in a `teal::init()` call.

### Decorating Module

This module generates the following objects, which can be modified in place using decorators:

- `plot` (`ggplot`)

A Decorator is applied to the specific output using a named list of `teal_transform_module` objects. The name of this list corresponds to the name of the output to which the decorator is applied. See code snippet below:

```
tm_g_gh_spaghettiplot(
   ..., # arguments for module
   decorators = list(
     plot = teal_transform_module(...) # applied only to `plot` output
   )
)
```

For additional details and examples of decorators, refer to the vignette `vignette("decorate-module-output", package = "teal.goshawk")`.

To learn more please refer to the vignette `vignette("transform-module-output", package = "teal")` or the `teal::teal_transform_module()` documentation.

### Reporting

This module returns an object of class `teal_module`, that contains a `server` function. Since the server function returns a `teal_report` object, this makes this module reportable, which means that the reporting functionality will be turned on automatically by the `teal` framework.

For more information on reporting in `teal`, see the vignettes:

- `vignette("reportable-shiny-application", package = "teal.reporter")`
- `vignette("adding-support-for-reporting-to-custom-modules", package = "teal")`

### Author(s)

Wenyi Liu

Balazs Toth

### 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"
)
ADSL <- teal.data::rADSL
ADLB <- teal.data::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")
})

join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_g_gh_spaghettiplot(
      label = "Spaghetti Plot",
      dataname = "ADLB",
      param = picks(
        variables("PARAMCD", "PARAMCD"),
        values(selected = "ALT", multiple = FALSE),
        check_dataset = FALSE
      ),
      idvar = "USUBJID",
      xaxis_var = variables(c("AVISITCD", "AVISIT"), "AVISITCD"),
      yaxis_var = variables(c("AVAL", "CHG", "PCHG"), "AVAL"),
      trt_group = variables(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


facet_wrap, *26*

init, *25*

maptrt, 3

shiny::helpText(), *6*, *11*, *17*, *21*, *26*

teal.picks::picks(), *5*, *10*, *16*, *20*, *25*
teal.picks::variables(), *5*, *10*, *16*, *20, 21*, *25*
teal.transform::choices_selected(), *5*, *10*, *16*, *20, 21*, *25*
teal::init(), *4*, *6*, *10, 11*, *16, 17*, *20*, *22*, *27*
teal::module(), *6*, *11*, *17*, *22*, *27*
teal::teal_transform_module(), *6*, *12*, *17*, *22*, *27*
tm_g_gh_boxplot, 4
tm_g_gh_correlationplot, 8
tm_g_gh_density_distribution_plot, 15
tm_g_gh_lineplot, 19
tm_g_gh_spaghettiplot, 24