Package: teal.modules.clinical (via r-universe)

October 14, 2026

Title 'teal' Modules for Standard Clinical Outputs

Version 0.14.0.9000

Date 2026-10-11

Description Provides user-friendly tools for creating and customizing clinical trial reports. By leveraging the 'teal' framework, this package provides 'teal' modules to easily create an interactive panel that allows for seamless adjustments to data presentation, thereby streamlining the creation of detailed and accurate reports.

License Apache License 2.0

URL https://insightsengineering.github.io/teal.modules.clinical/, https://github.com/insightsengineering/teal.modules.clinical/

BugReports https://github.com/insightsengineering/teal.modules.clinical/issues

Depends R (>= 4.4), teal (>= 1.2.0), teal.picks (>= 0.1.0), teal.transform (>= 0.7.2), tern (>= 0.9.9)

Imports broom (>= 0.7.10), bslib (>= 0.8.0), checkmate (>= 2.1.0), cowplot (>= 0.7.0), dplyr (>= 1.0.5), DT (>= 0.13), formatters (>= 0.5.11), ggplot2 (>= 3.4.0), ggrepel (>= 0.9.6), grid, lifecycle (>= 0.2.0), rlistings (>= 0.2.12), rmarkdown (>= 2.23), rtables (>= 0.6.13), scales (>= 1.4.0), shiny (>= 1.8.1), shinyjs (>= 1.10.0), shinyvalidate (>= 0.1.3), shinyWidgets (>= 0.5.1), stats, teal.code (>= 0.7.0), teal.data (>= 0.8.0), teal.logger (>= 0.4.0), teal.reporter (>= 0.6.0), teal.widgets (>= 0.5.0), tern.gee (>= 0.1.5), tern.mmrm (>= 0.3.4), tidyselect (>= 1.2.0), utils, vistime (>= 1.2.3)

Suggests forcats (>= 1.0.0), knitr (>= 1.42), logger (>= 0.4.0), lubridate (>= 1.7.9), nestcolor (>= 0.1.0), pkgload (>= 1.4.0), roxy.shinylive (>= 1.0.0), rvest (>= 1.0.4), shinytest2 (>= 0.4.1), styler (>= 1.10.3), testthat (>= 3.2.3), withr (>= 3.0.0)

VignetteBuilder knitr, rmarkdown

Config/Needs/verdepcheck insightsengineering/teal, insightsengineering/teal.slice, insightsengineering/teal.transform, insightsengineering/tern, tidymodels/broom, mllg/checkmate, wilkelab/cowplot, tidyverse/dplyr, rstudio/DT, insightsengineering/formatters, tidyverse/ggplot2, slowkow/ggrepel, r-lib/lifecycle, daroczig/logger, insightsengineering/rlistings, rstudio/rmarkdown, insightsengineering/rtables, r-lib/scales, rstudio/shiny, daattali/shinyjs, rstudio/shinyvalidate, dreamRs/shinyWidgets, insightsengineering/teal.code, insightsengineering/teal.data, insightsengineering/teal.logger, insightsengineering/teal.reporter, insightsengineering/teal.widgets, insightsengineering/tern.gee, insightsengineering/tern.mmrm, shosaco/vistime, tidyverse/forcats, yihui/knitr, tidyverse/lubridate, insightsengineering/nestcolor, r-lib/pkgload, insightsengineering/roxy.shinylive, tidyverse/rvest, rstudio/shinytest2, r-lib/styler, r-lib/testthat, r-lib/tidyselect, r-lib/withr

Config/Needs/website insightsengineering/nesttemplate

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE, packages = c(``roxy.shinylive''))

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 libx11-dev zlib1g-dev

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

Date/Publication 2026-10-11 20:35:29 UTC

RemoteUrl https://github.com/insightsengineering/teal.modules.clinical

RemoteRef HEAD

RemoteSha 8e3421093a3abd67b4de42179c3b1092fe8e62c1

Contents

teal.modules.clinical-package 4
add_expr 5
as_num 6
bracket_expr 7
call_concatenate 8
clean_description 9
color_lab_values 9

column_annotation_label . . . 10
cs_to_des_filter . . . 11
cs_to_des_select . . . 12
cs_to_filter_spec . . . 12
cs_to_select_spec . . . 13
default_total_label . . . 14
ex_data . . . 15
extract_input . . . 16
get_var_labels . . . 17
h_concat_expr . . . 17
is.cs_or_des . . . 18
pipe_expr . . . 19
prepare_arm . . . 19
prepare_arm_levels . . . 21
split_choices . . . 22
split_col_expr . . . 23
split_interactions . . . 24
tm_a_gee . . . 24
tm_a_mmrm . . . 28
tm_g_barchart_simple . . . 33
tm_g_ci . . . 38
tm_g_forest_rsp . . . 42
tm_g_forest_tte . . . 48
tm_g_ipp . . . 52
tm_g_km . . . 57
tm_g_lineplot . . . 62
tm_g_pp_adverse_events . . . 66
tm_g_pp_patient_timeline . . . 70
tm_g_pp_therapy . . . 74
tm_g_pp_vitals . . . 78
tm_t_abnormality . . . 81
tm_t_abnormality_by_worst_grade . . . 86
tm_t_ancova . . . 89
tm_t_binary_outcome . . . 94
tm_t_coxreg . . . 99
tm_t_events . . . 105
tm_t_events_by_grade . . . 109
tm_t_events_patyear . . . 113
tm_t_events_summary . . . 117
tm_t_exposure . . . 122
tm_t_glm_counts . . . 126
tm_t_logistic . . . 130
tm_t_mult_events . . . 134
tm_t_pp_basic_info . . . 138
tm_t_pp_laboratory . . . 140
tm_t_pp_medical_history . . . 142
tm_t_pp_prior_medication . . . 145
tm_t_shift_by_arm . . . 147

tm_t_shift_by_arm_by_worst 151
tm_t_shift_by_grade 155
tm_t_smq 159
tm_t_summary 162
tm_t_summary_by 166
tm_t_tte 170

Index **176**

`teal.modules.clinical-package`
`teal` *Modules for Standard Clinical Outputs*

Description

Provides teal modules for the standard clinical trials outputs. The teal modules add an encoding panel to interactively change the encodings within teal.

Author(s)

Maintainer: Dony Unardi <unardid@gene.com>

Authors:

- Dony Unardi <unardid@gene.com>
- Joe Zhu <joe.zhu@roche.com> (ORCID)
- Jana Stoilova <jana.stoilova@roche.com>
- Davide Garolini <davide.garolini@roche.com>
- Emily de la Rua <emily.de_la_rua@contractors.roche.com>
- Abinaya Yogasekaram <abinaya.yogasekaram@contractors.roche.com>
- Mahmoud Hallal <mahmoud.hallal@roche.com>
- Dawid Kaledkowski <dawid.kaledkowski@roche.com>
- Rosemary Li <li.yaqiong@gene.com>
- Heng Wang <wang.heng@gene.com>
- Pawel Rucki <pawel.rucki@roche.com>
- Nikolas Burkoff
- Konrad Pagacz

Other contributors:

- Vaakesan Sundrelingam [contributor]
- Francois Collin [contributor]
- Imanol Zubizarreta [contributor]
- F. Hoffmann-La Roche AG [copyright holder, funder]

See Also

Useful links:

- https://insightsengineering.github.io/teal.modules.clinical/
- https://github.com/insightsengineering/teal.modules.clinical/
- Report bugs at https://github.com/insightsengineering/teal.modules.clinical/issues

---

add_expr *Expression List*

Description

Add a new expression to a list (of expressions).

Usage

```
add_expr(expr_ls, new_expr)
```

Arguments

expr_ls	(list of call) the list to which a new expression should be added.
new_expr	(call) the new expression to add.

Details

Offers a stricter control to add new expressions to an existing list. The list of expressions can be later used to generate a pipeline, for instance with pipe_expr.

Value

a list of call.

Examples

```
library(rtables)

lyt <- list()
lyt <- add_expr(lyt, substitute(basic_table()))
lyt <- add_expr(
  lyt, substitute(split_cols_by(var = arm), env = list(armcd = "ARMCD"))
)
lyt <- add_expr(
  lyt,
  substitute(
    test_proportion_diff(
      vars = "rsp", method = "cmh", variables = list(strata = "strata")
```

```
    )
  )
)
lyt <- add_expr(lyt, quote(build_table(df = dta)))
pipe_expr(lyt)
```

---

as_num *Parse text input to numeric vector*

Description

Generic to parse text into numeric vectors. This was initially designed for a robust interpretation of text input in teal modules.

Usage

```
as_num(str)

## Default S3 method:
as_num(str)

## S3 method for class 'character'
as_num(str)

## S3 method for class 'numeric'
as_num(str)

## S3 method for class 'factor'
as_num(str)

## S3 method for class 'logical'
as_num(str)
```

Arguments

str	(vector) to extract numeric from.

Details

The function is intended to extract any numeric from a character string, factor levels, boolean and return a vector of numeric.

Value

As vector of numeric if directly parsed from `numeric` or boolean. A list of numeric if parsed from a character string, each character string associated with an list item.

Examples

```
dta <- list(
  character = c("text10,20.5letter30.!", "!-.40$$-50e5[", NA),
  factor    = factor(c("]+60e-6, 7.7%%8L", "%90sep.100\"1L", NA_character_)),
  numeric   = c(1, -5e+2, NA),
  logical   = c(TRUE, FALSE, NA)
)
lapply(dta, as_num)
```

`bracket_expr` *Expressions in Brackets*

Description

Groups several expressions in a single *bracketed* expression.

Usage

```
bracket_expr(exprs)
```

Arguments

`exprs`	(`list` of `call`) expressions to concatenate into a single *bracketed* expression.

Value

a { object. See `base::Paren()` for details.

Examples

```
adsl <- tmc_ex_adsl
adrs <- tmc_ex_adrs

expr1 <- substitute(
  expr = anl <- subset(df, PARAMCD == param),
  env = list(df = as.name("adrs"), param = "INVET")
)
expr2 <- substitute(expr = anl$rsp_lab <- tern::d_onco_rsp_label(anl$AVALC))
expr3 <- substitute(
  expr = {
    anl$is_rsp <- anl$rsp_lab %in%
      c("Complete Response (CR)", "Partial Response (PR)")
  }
)

res <- bracket_expr(list(expr1, expr2, expr3))
eval(res)
table(anl$rsp_lab, anl$is_rsp)
```

call_concatenate *Concatenate expressions via a binary operator*

Description

e.g. combine with + for `ggplot` without introducing parentheses due to associativity

Usage

```
call_concatenate(args, bin_op = "+")
```

Arguments

`args`	arguments to concatenate with operator
`bin_op`	binary operator to concatenate it with

Value

a `call`

Examples

```
library(ggplot2)

# What we want to achieve
call("+", quote(f), quote(g))
call("+", quote(f), call("+", quote(g), quote(h))) # parentheses not wanted
call("+", call("+", quote(f), quote(g)), quote(h)) # as expected without unnecessary parentheses
Reduce(function(existing, new) call("+", existing, new), list(quote(f), quote(g), quote(h)))

# how we do it
call_concatenate(list(quote(f), quote(g), quote(h)))
call_concatenate(list(quote(f)))
call_concatenate(list())
call_concatenate(
  list(quote(ggplot(mtcars)), quote(geom_point(aes(wt, mpg))))
)

eval(
  call_concatenate(
    list(quote(ggplot(mtcars)), quote(geom_point(aes(wt, mpg))))
  )
)
```

clean_description *Clean up categorical variable description*

Description

Cleaning categorical variable descriptions before presenting.

Usage

```
clean_description(x)
```

Arguments

x	(character) vector with categories descriptions.

Value

a string

Examples

```
clean_description("Level A (other text)")
clean_description("A long string that should be shortened")
```

color_lab_values *Mapping function for Laboratory Table*

Description

Map value and level characters to values with with proper html tags, colors and icons.

Usage

```
color_lab_values(
  x,
  classes = c("HIGH", "NORMAL", "LOW"),
  colors = list(HIGH = "red", NORMAL = "grey", LOW = "blue"),
  default_color = "black",
  icons = list(HIGH = "glyphicon glyphicon-arrow-up", LOW =
    "glyphicon glyphicon-arrow-down")
)
```

Arguments

`x`	(`character`) vector with elements under the format (`value level`).
`classes`	(`character`) classes vector.
`colors`	(`list`) color per class.
`default_color`	(`character`) default color.
`icons`	(`list`) certain icons per level.

Value

a character vector where each element is a formatted HTML tag corresponding to a value in `x`.

Examples

```
color_lab_values(c("LOW", "LOW", "HIGH", "NORMAL", "HIGH"))
```

---

column_annotation_label

Get full label, useful for annotating plots

Description

Get full label, useful for annotating plots

Usage

```
column_annotation_label(dataset, column, omit_raw_name = FALSE)
```

Arguments

`dataset`	(`data.frame`) dataset
`column`	(`character`) column to get label from
`omit_raw_name`	(`logical`) omits the raw name in square brackets if label is found

Value

"Label `[Column name]`" if label exists, otherwise "Column name".

Examples

```
data <- mtcars
column_annotation_label(data, "cyl")
attr(data[["cyl"]], "label") <- "Cylinder"
column_annotation_label(data, "cyl")
column_annotation_label(data, "cyl", omit_raw_name = TRUE)
column_annotation_label(tmc_ex_adsl, "ACTARM")
```

---

cs_to_des_filter *Convert choices_selected to data_extract_spec with only filter_spec*

Description

Convert choices_selected to data_extract_spec with only filter_spec

Usage

```
cs_to_des_filter(
  cs,
  dataname,
  multiple = FALSE,
  include_vars = FALSE,
  label = "Filter by"
)
```

Arguments

cs	(choices_selected) object to be transformed. See teal.transform::choices_selected() for details.
dataname	(character) name of the data
multiple	(logical) Whether multiple values shall be allowed in the shiny shiny::selectInput().
include_vars	(flag) whether to include the filter variables as fixed selection in the result. This can be useful for preserving for reuse in rtables code e.g.
label	(character) Label to print over the selection field. For no label, set to NULL.

Value

(teal.transform::data_extract_spec())

cs_to_des_select *Convert choices_selected to data_extract_spec with only select_spec*

Description

Convert choices_selected to data_extract_spec with only select_spec

Usage

```
cs_to_des_select(
  cs,
  dataname,
  multiple = FALSE,
  ordered = FALSE,
  label = "Select"
)
```

Arguments

cs	(choices_selected) object to be transformed. See teal.transform::choices_selected() for details.
dataname	(character) name of the data
multiple	(logical) Whether multiple values shall be allowed in the shiny shiny::selectInput().
ordered	(logical(1)) Flags whether selection order should be tracked.
label	(character) Label to print over the selection field. For no label, set to NULL.

Value

(teal.transform::data_extract_spec())

cs_to_filter_spec *Convert choices_selected to filter_spec*

Description

Convert choices_selected to filter_spec

Usage

```
cs_to_filter_spec(cs, multiple = FALSE, label = "Filter by")
```

Arguments

cs	(choices_selected) object to be transformed. See teal.transform::choices_selected() for details.
multiple	(logical) Whether multiple values shall be allowed in the shiny shiny::selectInput().
label	(character) Label to print over the selection field. For no label, set to NULL.

Value

(teal.transform::filter_spec())

---

cs_to_select_spec *Convert choices_selected to select_spec*

Description

Convert choices_selected to select_spec

Usage

```
cs_to_select_spec(cs, multiple = FALSE, ordered = FALSE, label = "Select")
```

Arguments

cs	(choices_selected) object to be transformed. See teal.transform::choices_selected() for details.
multiple	(logical) Whether multiple values shall be allowed in the shiny shiny::selectInput().
ordered	(logical(1)) Flags whether selection order should be tracked.
label	(character) Label to print over the selection field. For no label, set to NULL.

Value

(select_spec)

default_total_label *Default string for total column label*

Description

The default string used as a label for the "total" column. This value is used as the default value for the `total_label` argument throughout the `teal.modules.clinical` package. If not specified for each module by the user via the `total_label` argument, or in the R environment options via `set_default_total_label()`, then `"All Patients"` is used.

Usage

```
default_total_label()

set_default_total_label(total_label)
```

Arguments

`total_label`	(`string`) Single string value to set in the R environment options as the default label to use for the "total" column. Use `getOption("tmc_default_total_label")` to check the current value set in the R environment (defaults to `"All Patients"` if not set).

Value

- `default_total_label` returns the current value if an R environment option has been set for `"tmc_default_total_label"`, or `"All Patients"` otherwise.
- `set_default_total_label` has no return value.

Functions

- `default_total_label()`: Getter for default total column label.
- `set_default_total_label()`: Setter for default total column label. Sets the option `"tmc_default_total_label"` within the R environment.

Examples

```
# Default settings
default_total_label()
getOption("tmc_default_total_label")

# Set custom value
set_default_total_label("All Patients")

# Settings after value has been set
default_total_label()
```

```
getOption("tmc_default_total_label")
```

---

ex_data *Simulated* CDISC *Data for Examples*

Description

Simulated CDISC Data for Examples

Usage

```
tmc_ex_adsl

tmc_ex_adae

tmc_ex_adaette

tmc_ex_adcm

tmc_ex_adeg

tmc_ex_adex

tmc_ex_adlb

tmc_ex_admh

tmc_ex_adqs

tmc_ex_adrs

tmc_ex_adtte

tmc_ex_advs
```

Format

rds (data.frame)

An object of class tbl_df (inherits from tbl, data.frame) with 200 rows and 26 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 541 rows and 51 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 1800 rows and 35 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 512 rows and 45 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 5200 rows and 48 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 200 rows and 37 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 3000 rows and 58 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 1077 rows and 33 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 7000 rows and 36 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 1600 rows and 34 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 1000 rows and 34 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 8400 rows and 34 columns.

Functions

- tmc_ex_adsl: ADSL data
- tmc_ex_adae: ADAE data
- tmc_ex_adaette: ADAETTE data
- tmc_ex_adcm: ADCM data
- tmc_ex_adeg: ADEG data
- tmc_ex_adex: ADEX data
- tmc_ex_adlb: ADLB data
- tmc_ex_admh: ADMH data
- tmc_ex_adqs: ADQS data
- tmc_ex_adrs: ADRS data
- tmc_ex_adtte: ADTTE data
- tmc_ex_advs: ADVS data

---

extract_input *Extracts html id for* data_extract_ui

Description

The data_extract_ui is located under extended html id. We could not use ns("original id") for reference, as it is extended with specific suffixes.

Usage

```
extract_input(varname, dataname, filter = FALSE)
```

Arguments

varname	(character) the original html id. This should be retrieved with ns("original id") in the UI function or session$ns("original id")/"original id" in the server function.
dataname	(character) dataname from data_extract input. This might be retrieved like data_extract_spec(...)[[1]]$datana
filter	(logical) optional, if the connected extract_data_spec has objects passed to its filter argument

Value

a string

Examples

```
extract_input("ARM", "ADSL")
```

---

get_var_labels *Get variable labels*

Description

[Deprecated]

Usage

```
get_var_labels(datasets, dataname, vars)
```

Arguments

datasets	(teal::FilteredData) Data built up by teal
dataname	(character) name of the dataset
vars	(character) Column names in the data

Value

character variable labels.

---

h_concat_expr *Expression Deparsing*

Description

Deparse an expression into a string.

Usage

```
h_concat_expr(expr)
```

Arguments

expr	(call) or an object which can be used as so.

Value

a string.

Examples

```
expr <- quote({
  library(rtables)
  basic_table() %>%
    split_cols_by(var = "ARMCD") %>%
    test_proportion_diff(
      vars = "rsp", method = "cmh", variables = list(strata = "strata")
    ) %>%
    build_table(df = dta)
})

h_concat_expr(expr)
```

---

is.cs_or_des — *Whether object is of class* teal.transform::choices_selected()

Description

Whether object is of class teal.transform::choices_selected()

Usage

```
is.cs_or_des(x)
```

Arguments

x	object to be checked

Value

(logical)

pipe_expr — *Expressions as a Pipeline*

Description

Concatenate expressions in a single pipeline-flavor expression.

Usage

```
pipe_expr(exprs, pipe_str = "%>%")
```

Arguments

exprs	(list of call) expressions to concatenate in a pipeline (%>%).
pipe_str	(character) the character which separates the expressions.

Value

a call

Examples

```
pipe_expr(
  list(
    expr1 = substitute(df),
    expr2 = substitute(head)
  )
)
```

prepare_arm — *Expression: Arm Preparation*

Description

The function generate the standard expression for pre-processing of dataset in teal module applications. This is especially of interest when the same preprocessing steps needs to be applied similarly to several datasets (e.g. ADSL and ADRS).

Usage

```
prepare_arm(
  dataname,
  arm_var,
  ref_arm,
  comp_arm,
  compare_arm = !is.null(ref_arm),
  ref_arm_val = paste(ref_arm, collapse = "/"),
  drop = TRUE
)
```

Arguments

`dataname`	(`character`) analysis data used in teal module.
`arm_var`	(`character`) variable names that can be used as `arm_var`.
`ref_arm`	(`character`) the level of reference arm in case of arm comparison.
`comp_arm`	(`character`) the level of comparison arm in case of arm comparison.
`compare_arm`	(`logical`) triggers the comparison between study arms.
`ref_arm_val`	(`character`) replacement name for the reference level.
`drop`	(`logical`) drop the unused variable levels.

Details

In `teal.modules.clinical`, the user interface includes manipulation of the study arms. Classically: the arm variable itself (e.g. `ARM`, `ACTARM`), the reference arm (0 or more), the comparison arm (1 or more) and the possibility to combine comparison arms.

Note that when no arms should be compared with each other, then the produced expression is reduced to optionally dropping non-represented levels of the arm.

When comparing arms, the pre-processing includes three steps:

1. Filtering of the dataset to retain only the arms of interest (reference and comparison).
2. Optional, if more than one arm is designated as *reference* they are combined into a single level.
3. The reference is explicitly reassigned and the non-represented levels of arm are dropped.

Value

a `call`

Examples

```
prepare_arm(
  dataname = "adrs",
  arm_var = "ARMCD",
  ref_arm = "ARM A",
  comp_arm = c("ARM B", "ARM C")
)

prepare_arm(
  dataname = "adsl",
  arm_var = "ARMCD",
  ref_arm = c("ARM B", "ARM C"),
  comp_arm = "ARM A"
)
```

---

prepare_arm_levels *Expression: Prepare Arm Levels*

Description

This function generates the standard expression for pre-processing of dataset arm levels in and is used to apply the same steps in safety teal modules.

Usage

```
prepare_arm_levels(dataname, parentname, arm_var, drop_arm_levels = TRUE)
```

Arguments

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (character)
variable names that can be used as `arm_var`.

`drop_arm_levels` (logical)
whether to drop unused levels of `arm_var`. If `TRUE`, `arm_var` levels are set to those used in the `dataname` dataset. If `FALSE`, `arm_var` levels are set to those used in the `parentname` dataset. If `dataname` and `parentname` are the same, then `drop_arm_levels` is set to `TRUE` and user input for this parameter is ignored.

Value

a { object. See `base::Paren()` for details.

Examples

```
prepare_arm_levels(
  dataname = "adae",
  parentname = "adsl",
  arm_var = "ARMCD",
  drop_arm_levels = TRUE
)

prepare_arm_levels(
  dataname = "adae",
  parentname = "adsl",
  arm_var = "ARMCD",
  drop_arm_levels = FALSE
)
```

---

split_choices — *Split* `choices_selected` *objects with interactions into their component variables*

Description

Split `choices_selected` objects with interactions into their component variables

Usage

```
split_choices(x)
```

Arguments

`x`	(`choices_selected`) object with interaction terms

Value

a `teal.transform::choices_selected()` object. When `x` is a `delayed_choices_selected` object (created with delayed data), it is returned unchanged because the actual choices are not available until data is resolved at runtime.

Note

uses the regex `\\*|:` to perform the split.

Examples

```
split_choices(choices_selected(choices = c("x:y", "a*b"), selected = all_choices()))

# Also works with delayed data - returns the object unchanged
split_choices(choices_selected(variable_choices("ADSL")))
```

split_col_expr *Split-Column Expression*

Description

Renders the expression for column split in `rtables` depending on:

- the expected or not arm comparison
- the expected or not arm combination

Usage

```
split_col_expr(compare, combine, ref, arm_var)
```

Arguments

`compare`	(`logical`) if `TRUE` the reference level is included.
`combine`	(`logical`) if `TRUE` the group combination is included.
`ref`	(`character`) the reference level (not used for `combine = TRUE`).
`arm_var`	(`character`) the arm or grouping variable name.

Value

a `call`

Examples

```
split_col_expr(
  compare = TRUE,
  combine = FALSE,
  ref = "ARM A",
  arm_var = "ARMCD"
)
```

split_interactions *Split interaction terms into their component variables*

Description

Split interaction terms into their component variables

Usage

```
split_interactions(x, by = "\\*|:")
```

Arguments

x	(character) string representing the interaction usually in the form `x:y` or `x*y`.
by	(character) regex with which to split the interaction term by.

Value

a vector of strings where each element is a component variable extracted from interaction term `x`.

Examples

```
split_interactions("x:y")
split_interactions("x*y")
```

tm_a_gee *teal Module: Generalized Estimating Equations (GEE) analysis*

Description

This module produces an analysis table using Generalized Estimating Equations (GEE).

Usage

```
tm_a_gee(
  label,
  dataname,
  parentname = "ADSL",
  aval_var,
  id_var,
  arm_var,
  visit_var,
  cov_var,
```

```
  arm_ref_comp = NULL,
  paramcd,
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`aval_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

`id_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object specifying the variable name for subject id.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `arm_var`. It defines the grouping variable in the results table.

`visit_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `visit` variable. Must be a factor in `dataname`.

`cov_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the covariates variables.

`arm_ref_comp` (`list`) optional,
if specified it must be a named list with each element corresponding to an arm variable in `ADSL` and the element must be another list (possibly with delayed `teal.picks::variables()` or `teal.picks::values()`; legacy `teal.transform::variable_choices` and `teal.transform::value_choices()` are deprecated but still accepted) with the elements named `ref` and `comp` that define the default reference and comparison arms when the arm variable is changed.

`paramcd` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the parameter code variable from `dataname`.

conf_level (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
available confidence levels and default selection, each in the range (0, 1).

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.

basic_table_args
(basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- table (ElementaryTable - output of rtables::build_table())

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_a_gee(
   ..., # arguments for module
   decorators = list(
     table = teal_transform_module(...) # applied to the `table` output
   )
)
```

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  library(teal.modules.clinical)
  library(dplyr)
  ADSL <- tmc_ex_adsl
  ADQS <- tmc_ex_adqs %>%
    filter(ABLFL != "Y" & ABLFL2 != "Y") %>%
    mutate(
      AVISIT = as.factor(AVISIT),
      AVISITN = rank(AVISITN) %>%
        as.factor() %>%
        as.numeric() %>%
        as.factor(),
      AVALBIN = AVAL < 50 # Just as an example to get a binary endpoint.
    ) %>%
    droplevels()
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

all_values <- function(x) unique(x)
class(all_values) <- append(class(all_values), "des-delayed")

app <- init(
  data = data,
  modules = modules(
    tm_a_gee(
      label = "GEE",
      dataname = "ADQS",
      aval_var = variables(choices = "AVALBIN"),
      id_var = variables(choices = c("USUBJID", "SUBJID"), selected = "USUBJID"),
      arm_var = variables(choices = c("ARM", "ARMCD"), selected = "ARM"),
      visit_var = variables(choices = c("AVISIT", "AVISITN"), selected = "AVISIT"),
```

```
      paramcd = picks(
        variables(choices = c("PARAMCD", "PARAM")),
        values(all_values, "FKSI-FWB"),
        check_dataset = FALSE
      ),
      cov_var = variables(
        choices = c("BASE", "AGE", "SEX", "BASE:AVISIT"),
        selected = NULL
      )
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_a_mmrm *teal Module: Mixed Model Repeated Measurements (MMRM) Analysis*

Description

This module produces analysis tables and plots for Mixed Model Repeated Measurements.

Usage

```
tm_a_mmrm(
  label,
  dataname,
  parentname = "ADSL",
  aval_var,
  id_var,
  arm_var,
  visit_var,
  cov_var,
  arm_ref_comp = NULL,
  paramcd,
  method = teal.picks::values(c("Satterthwaite", "Kenward-Roger",
    "Kenward-Roger-Linear"), "Satterthwaite"),
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  plot_height = c(700L, 200L, 2000L),
  plot_width = NULL,
  total_label = default_total_label(),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  ggplot2_args = teal.widgets::ggplot2_args(),
```

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

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.
parentname	(character) parent analysis data used in teal module, usually this refers to ADSL.
aval_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and pre-selected option for the analysis variable.
id_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object specifying the variable name for subject id.
arm_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable in the results table.
visit_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for variable names that can be used as visit variable. Must be a factor in dataname.
cov_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for the covariates variables.
arm_ref_comp	(list) optional, if specified it must be a named list with each element corresponding to an arm variable in ADSL and the element must be another list (possibly with delayed teal.picks::variables() or teal.picks::values(); legacy teal.transform::variable_choices and teal.transform::value_choices() are deprecated but still accepted) with the elements named ref and comp that define the default reference and comparison arms when the arm variable is changed.
paramcd	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for the parameter code variable from dataname.
method	(teal.picks::values or teal.transform::choices_selected()) object with all available choices and pre-selected option for the adjustment method.
conf_level	(teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted) available confidence levels and default selection, each in the range (0, 1).

`plot_height` (numeric) optional
vector of length three with `c(value, min, max)`. Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.

`plot_width` (numeric) optional
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.

`total_label` (string)
string to display as total column/row label if column/row is enabled (see `add_total`). Defaults to `"All Patients"`. To set a new default `total_label` to apply in all modules, run `set_default_total_label("new_default")`.

`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.

`basic_table_args`
(`basic_table_args`) optional
object created by `teal.widgets::basic_table_args()` with settings for the module table. The argument is merged with option `teal.basic_table_args` and with default module arguments (hard coded in the module body). For more details, see the vignette: `vignette("custom-basic-table-arguments", package = "teal.widgets")`.

`ggplot2_args` (`ggplot2_args`) optional
object created by `teal.widgets::ggplot2_args()` with settings for all the plots or named list of `ggplot2_args` objects for plot-specific settings. List names should match the following: `c("default", "lsmeans", "diagnostic")`. The argument is merged with option `teal.ggplot2_args` and with default module arguments (hard coded in the module body). For more details, see the help vignette: `vignette("custom-ggplot2-arguments", package = "teal.widgets")`.

`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.

Decorating Module

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

- `lsmeans_plot` (`ggplot`)

- diagnostic_plot (ggplot)
- lsmeans_table (TableTree- output from rtables::build_table)
- covariance_table (ElementaryTable- output from rtables::build_table)
- fixed_effects_table (ElementaryTable- output from rtables::build_table)
- diagnostic_table (ElementaryTable- output from rtables::build_table)

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_a_mrmm(
   ..., # arguments for module
   decorators = list(
    lsmeans_plot = teal_transform_module(...), # applied only to `lsmeans_plot` output
    diagnostic_plot = teal_transform_module(...), # applied only to `diagnostic_plot` output
    lsmeans_table = teal_transform_module(...), # applied only to `lsmeans_table` output
    covariance_table = teal_transform_module(...), # applied only to `covariance_table` output
    fixed_effects_table = teal_transform_module(...), # applied only to `fixed_effects_table` output
    diagnostic_table = teal_transform_module(...) # applied only to `diagnostic_table` output
   )
)
```

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

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")

Examples in Shinylive

example-1 Open in Shinylive

Note

The ordering of the input data sets can lead to slightly different numerical results or different convergence behavior. This is a known observation with the used package lme4. However, once convergence is achieved, the results are reliable up to numerical precision.

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
arm_ref_comp <- list(
  ARMCD = list(
    ref = "ARM B",
    comp = c("ARM A", "ARM C")
  )
)

data <- teal_data()
data <- within(data, {
  library(teal.modules.clinical)
  library(dplyr)
  ADSL <- tmc_ex_adsl
  ADQS <- tmc_ex_adqs %>%
    filter(ABLFL != "Y" & ABLFL2 != "Y") %>%
    filter(AVISIT %in% c("WEEK 1 DAY 8", "WEEK 2 DAY 15", "WEEK 3 DAY 22")) %>%
    mutate(
      AVISIT = as.factor(AVISIT),
      AVISITN = rank(AVISITN) %>%
        as.factor() %>%
        as.numeric() %>%
        as.factor() # making consecutive numeric factor
    )
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_a_mmrm(
      label = "MMRM",
      dataname = "ADQS",
      aval_var = variables(c("AVAL", "CHG"), multiple = FALSE),
      id_var = variables(c("USUBJID", "SUBJID"), multiple = FALSE),
      arm_var = variables(c("ARM", "ARMCD")),
      visit_var = variables(c("AVISIT", "AVISITN")),
      arm_ref_comp = arm_ref_comp,
      paramcd = picks(
        variables("PARAMCD", "PARAMCD"),
        values(selected = "FKSI-FWB", multiple = FALSE),
        check_dataset = FALSE
      ),
    cov_var = variables(c("BASE", "AGE", "SEX", interaction_vars("BASE", "AVISIT")), NULL)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
```

```
}
```

---

tm_g_barchart_simple *teal Module: Simple Bar Chart and Table of Counts per Category*

Description

This module produces a `ggplot2::ggplot()` type bar chart and summary table of counts per category.

Usage

```
tm_g_barchart_simple(
  x = NULL,
  fill = NULL,
  x_facet = NULL,
  y_facet = NULL,
  label = "Count Barchart",
  plot_options = NULL,
  plot_height = c(600L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)

## Default S3 method:
tm_g_barchart_simple(
  x = NULL,
  fill = NULL,
  x_facet = NULL,
  y_facet = NULL,
  label = "Count Barchart",
  plot_options = NULL,
  plot_height = c(600L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)

## S3 method for class 'picks'
```

```
tm_g_barchart_simple(
  x = NULL,
  fill = NULL,
  x_facet = NULL,
  y_facet = NULL,
  label = "Count Barchart",
  plot_options = NULL,
  plot_height = c(600L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

x	(picks, data_extract_spec, or list of data_extract_spec) variable on the x-axis (required).
fill	(NULL, picks, data_extract_spec, or list thereof) grouping variable for bar colors.
x_facet	(NULL, picks, data_extract_spec, or list thereof) column-wise faceting groups.
y_facet	(NULL, picks, data_extract_spec, or list thereof) row-wise faceting groups.
label	(character) menu item label of the module in the teal app.
plot_options	(list) list of plot options.
plot_height	(numeric) optional vector of length three with c(value, min, max). Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.
plot_width	(numeric) optional 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.
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.
ggplot2_args	(ggplot2_args) optional object created by teal.widgets::ggplot2_args() with settings for the module plot. The argument is merged with option teal.ggplot2_args and with

default module arguments (hard coded in the module body). For more details, see the vignette: `vignette("custom-ggplot2-arguments", package = "teal.widgets")`.

`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.

Details

Categories can be defined up to four levels deep through `x`, `fill`, `x_facet`, and `y_facet`. The `x` encoding is always required; `fill`, `x_facet`, and `y_facet` may be `NULL` to omit them.

S3 dispatch uses the class of `x`: `tm_g_barchart_simple.default()` for `teal.transform::data_extract_spec()` (or `list` thereof) and `tm_g_barchart_simple.picks()` for `teal.picks::picks()`. Do not mix `data_extract_spec` and `picks` encodings in one call.

Value

a `teal_module` object.

Methods (by class)

- `tm_g_barchart_simple(default)`: Legacy `teal.transform::data_extract_spec()` encodings.
- `tm_g_barchart_simple(picks)`: `teal.picks::picks()`-based encodings (`picks`).

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_barchart_simple(
   ..., # 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.modules.clinical")`.

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(nestcolor)

data <- teal_data()
data <- within(data, {
  library(teal.modules.clinical)
  library(formatters)
  library(dplyr)
  ADSL <- tmc_ex_adsl %>%
    mutate(ITTFL = factor("Y") %>%
      with_label("Intent-To-Treat Population Flag"))
  ADAE <- tmc_ex_adae %>%
    filter(!((AETOXGR == 1) & (AESEV == "MILD") & (ARM == "A: Drug X")))
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADAE <- data[["ADAE"]]

app <- init(
  data = data,
  modules = modules(
    tm_g_barchart_simple(
      label = "ADAE Analysis",
      x = picks(
        datasets("ADSL"),
        variables(
          choices = c(
            "ARM", "ACTARM", "SEX",
            "RACE", "ITTFL", "SAFFL", "STRATA2"
          ),
          selected = "ACTARM",
          multiple = FALSE
```

```
          )
        ),
        fill = picks(
          datasets(choices = c("ADSL", "ADAE")),
          variables(
            choices = c(
              "ARM", "ACTARM", "SEX",
              "RACE", "ITTFL", "SAFFL", "STRATA2",
              "AETOXGR", "AESEV", "AESER"
            ),
            selected = "SEX",
            multiple = FALSE
          )
        ),
        x_facet = picks(
          datasets(choices = c("ADAE", "ADSL")),
          variables(
            choices = c(
              "AETOXGR", "AESEV", "AESER",
              "ARM", "ACTARM", "SEX",
              "RACE", "ITTFL", "SAFFL", "STRATA2"
            ),
            selected = "AETOXGR",
            multiple = FALSE
          )
        ),
        y_facet = picks(
          datasets(choices = c("ADAE", "ADSL")),
          variables(
            choices = c(
              "AETOXGR", "AESEV", "AESER",
              "ARM", "ACTARM", "SEX",
              "RACE", "ITTFL", "SAFFL", "STRATA2"
            ),
            selected = "AESEV",
            multiple = FALSE
          )
        )
      )
    )
  )
if (interactive()) {
  shinyApp(app$ui, app$server)
}

# Legacy `teal.transform::data_extract_spec()` encodings (default S3 method):
## Not run:
data <- teal_data()
data <- within(data, {
  library(dplyr)
  ADSL <- tmc_ex_adsl %>%
    mutate(ITTFL = factor("Y") %>% with_label("Intent-To-Treat Population Flag"))
})
```

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

app <- init(
  data = data,
  modules = modules(
    tm_g_barchart_simple(
      x = teal.transform::data_extract_spec(
        dataname = "ADSL",
        select = teal.transform::select_spec(
          choices = teal.transform::variable_choices("ADSL", c("ARM", "SEX")),
          selected = "ARM",
          multiple = FALSE
        )
      )
    )
  )
)
shiny::shinyApp(app$ui, app$server)

## End(Not run)
```

---

tm_g_ci *teal Module: Confidence Interval Plot*

Description

This module produces a `ggplot2::ggplot()` type confidence interval plot consistent with the TLG Catalog template `CIG01` available here.

Usage

```
tm_g_ci(
  label,
  x_var,
  y_var,
  paramcd = NULL,
  avisit = NULL,
  color,
  stat = c("mean", "median"),
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  plot_height = c(700L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

```
## S3 method for class 'data_extract_spec'
tm_g_ci(
  label,
  x_var,
  y_var,
  paramcd = NULL,
  avisit = NULL,
  color,
  stat = c("mean", "median"),
 conf_level = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
    TRUE),
  plot_height = c(700L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)

## Default S3 method:
tm_g_ci(
  label,
  x_var,
  y_var,
  paramcd,
  avisit,
  color,
  stat = c("mean", "median"),
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  plot_height = c(700L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
x_var	(teal.picks::variables(), teal.picks::picks(), or legacy data_extract_spec) treatment-axis encoding.
y_var	(teal.picks::variables(), teal.picks::picks(), or legacy data_extract_spec) analysis-value encoding.

paramcd
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the parameter code variable from dataname.

avisit
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
value of analysis visit AVISIT of interest.

color
: (teal.picks::variables(), teal.picks::picks(), or legacy data_extract_spec)
grouping variable for colors, shapes, and line types.

stat
: (character)
statistic to plot. Options are "mean" and "median".

conf_level
: (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
available confidence levels and default selection, each in the range (0, 1).

plot_height
: (numeric) optional
vector of length three with c(value, min, max). Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.

plot_width
: (numeric) optional
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.

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.

ggplot2_args
: (ggplot2_args) optional
object created by teal.widgets::ggplot2_args() with settings for the module plot. The argument is merged with option teal.ggplot2_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-ggplot2-arguments", package = "teal.widgets").

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.

Methods (by class)

- `tm_g_ci(data_extract_spec)`: Legacy encodings via `data_extract_spec` (merge-based UI).
- `tm_g_ci(default)`: teal.picks encodings via `picks` objects for `x_var`, `y_var`, and `color` (use `tm_g_ci()` to pass `teal.picks::variables()` objects; they are wrapped into `picks`).

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_ci(
   ..., # 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.modules.clinical")`.

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(nestcolor)

data <- teal_data()
data <- within(data, {
  library(dplyr)
  ADSL <- tmc_ex_adsl
  ADLB <- tmc_ex_adlb
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_g_ci(
      label = "Confidence Interval Plot",
      x_var = picks(
        datasets("ADSL", "ADSL"),
        variables(c("ARMCD", "BMRKR2"), "ARMCD")
      ),
      y_var = picks(
        datasets("ADLB", "ADLB"),
        variables(c("AVAL", "CHG"), "AVAL")
      ),
      color = picks(
        datasets("ADSL", "ADSL"),
        variables(c("SEX", "STRATA1", "STRATA2"), "STRATA1")
      ),
      paramcd = picks(
        datasets("ADLB", "ADLB"),
        variables("PARAMCD", "PARAMCD"),
        values(selected = "ALT", multiple = FALSE)
      ),
      avisit = picks(
        datasets("ADLB", "ADLB"),
        variables("AVISIT", "AVISIT"),
        values(selected = "SCREENING", multiple = FALSE)
      )
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_g_forest_rsp` *teal Module: Forest Response Plot*

Description

This module produces a grid-style forest plot for response data with ADaM structure.

Usage

```
tm_g_forest_rsp(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  arm_ref_comp = NULL,
  paramcd,
  aval_var = teal.picks::variables("AVALC", "AVALC", fixed = TRUE),
  subgroup_var,
  strata_var,
  stats = c("n_tot", "n", "n_rsp", "prop", "or", "ci"),
  riskdiff = NULL,
  fixed_symbol_size = TRUE,
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  default_responses = c("CR", "PR", "Y", "Complete Response (CR)",
    "Partial Response (PR)"),
  plot_height = c(500L, 200L, 2000L),
  plot_width = c(1500L, 800L, 3000L),
  rel_width_forest = c(25L, 0L, 100L),
  font_size = c(15L, 1L, 30L),
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `arm_var`. It defines the grouping variable in the results table.

`arm_ref_comp` (`list`) optional,
if specified it must be a named list with each element corresponding to an arm variable in `ADSL` and the element must be another list (possibly with delayed

teal.picks::variables() or teal.picks::values(); legacy teal.transform::variable_choices and teal.transform::value_choices() are deprecated but still accepted) with the elements named ref and comp that define the default reference and comparison arms when the arm variable is changed.

paramcd
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the parameter code variable from dataname.

aval_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

subgroup_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as the default subgroups.

strata_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
names of the variables for stratified analysis.

stats
: (character)
the names of statistics to be reported among:

- n: Total number of observations per group.
- n_rsp: Number of responders per group.
- prop: Proportion of responders.
- n_tot: Total number of observations.
- or: Odds ratio.
- ci : Confidence interval of odds ratio.
- pval: p-value of the effect. Note, the statistics n_tot, or, and ci are required.

riskdiff
: (list)
if a risk (proportion) difference column should be added, a list of settings to apply within the column. See tern::control_riskdiff() for details. If NULL, no risk difference column will be added.

fixed_symbol_size
: (logical)
When (TRUE), the same symbol size is used for plotting each estimate. Otherwise, the symbol size will be proportional to the sample size in each each subgroup.

conf_level
: (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
available confidence levels and default selection, each in the range (0, 1).

default_responses
: (list or character)
defines the default codes for the response variable in the module per value of paramcd. A passed vector is transmitted for all paramcd values. A passed list must be named and contain arrays, each name corresponding to a single value of

paramcd. Each array may contain default response values or named arrays rsp of default selected response values and levels of default level choices.

plot_height (numeric) optional
vector of length three with c(value, min, max). Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.

plot_width (numeric) optional
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.

rel_width_forest (proportion)
proportion of total width to allocate to the forest plot. Relative width of table is then 1 - rel_width_forest. If as_list = TRUE, this parameter is ignored.

font_size (numeric(1))
font size.

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.

ggplot2_args (ggplot2_args) optional
object created by teal.widgets::ggplot2_args() with settings for the module plot. For this module, this argument will only accept ggplot2_args object with labs list of following child elements: title, caption. No other elements would be taken into account. The argument is merged with option teal.ggplot2_args and with default module arguments (hard coded in the module body).

For more details, see the vignette: vignette("custom-ggplot2-arguments", package = "teal.widgets").

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.

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_forest_rsp(
   ..., # 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.modules.clinical")`.

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(nestcolor)

data <- teal_data()
data <- within(data, {
  library(formatters)
  library(dplyr)
  ADSL <- teal.modules.clinical::tmc_ex_adsl
  ADRS <- teal.modules.clinical::tmc_ex_adrs %>%
    mutate(AVALC = d_onco_rsp_label(AVALC) %>%
      with_label("Character Result/Finding")) %>%
    filter(PARAMCD != "OVRINV" | AVISIT == "FOLLOW UP")
})
join_keys(data) <- default_cdisc_join_keys[names(data)]
```

```
arm_ref_comp <- list(
  ARM = list(
    ref = "B: Placebo",
    comp = c("A: Drug X", "C: Combination")
  ),
  ARMCD = list(
    ref = "ARM B",
    comp = c("ARM A", "ARM C")
  )
)

app <- init(
  data = data,
  modules = modules(
    tm_g_forest_rsp(
      label = "Forest Response",
      dataname = "ADRS",
      arm_var = variables(c("ARM", "ARMCD"), "ARMCD"),
      arm_ref_comp = arm_ref_comp,
      paramcd = picks(
        variables("PARAMCD", "PARAMCD"),
        values(selected = "INVET", multiple = FALSE),
        check_dataset = FALSE
      ),
      subgroup_var = variables(c("BMRKR2", "SEX"), "BMRKR2", multiple = TRUE),
      strata_var = variables(c("STRATA1", "STRATA2"), "STRATA2", multiple = TRUE),
      plot_height = c(600L, 200L, 2000L),
      default_responses = list(
        BESRSPI = list(
          rsp = c("Stable Disease (SD)", "Not Evaluable (NE)"),
          levels = c(
            "Complete Response (CR)", "Partial Response (PR)", "Stable Disease (SD)",
            "Progressive Disease (PD)", "Not Evaluable (NE)"
          )
        ),
        INVET = list(
          rsp = c("Complete Response (CR)", "Partial Response (PR)"),
          levels = c(
            "Complete Response (CR)", "Not Evaluable (NE)", "Partial Response (PR)",
            "Progressive Disease (PD)", "Stable Disease (SD)"
          )
        ),
        OVRINV = list(
          rsp = c("Progressive Disease (PD)", "Stable Disease (SD)"),
        levels = c("Progressive Disease (PD)", "Stable Disease (SD)", "Not Evaluable (NE)")
        )
      )
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

tm_g_forest_tte *teal Module: Forest Survival Plot*

Description

This module produces a grid-style forest plot for time-to-event data with ADaM structure.

Usage

```
tm_g_forest_tte(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  arm_ref_comp = NULL,
  subgroup_var,
  paramcd,
  strata_var,
  aval_var = teal.picks::variables("AVAL", "AVAL", fixed = TRUE),
  cnsr_var = teal.picks::variables("CNSR", "CNSR", fixed = TRUE),
  stats = c("n_tot_events", "n_events", "median", "hr", "ci"),
  riskdiff = NULL,
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  time_unit_var = teal.picks::variables("AVALU", "AVALU", fixed = TRUE),
  fixed_symbol_size = TRUE,
  plot_height = c(500L, 200L, 2000L),
  plot_width = c(1500L, 800L, 3000L),
  rel_width_forest = c(25L, 0L, 100L),
  font_size = c(15L, 1L, 30L),
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label`	(character) menu item label of the module in the teal app.
`dataname`	(character) analysis data used in teal module.
`parentname`	(character) parent analysis data used in teal module, usually this refers to `ADSL`.

arm_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable in the results table.

arm_ref_comp
(list) optional,
if specified it must be a named list with each element corresponding to an arm variable in ADSL and the element must be another list (possibly with delayed teal.picks::variables() or teal.picks::values(); legacy teal.transform::variable_choices and teal.transform::value_choices() are deprecated but still accepted) with the elements named ref and comp that define the default reference and comparison arms when the arm variable is changed.

subgroup_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as the default subgroups.

paramcd
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the parameter code variable from dataname.

strata_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
names of the variables for stratified analysis.

aval_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

cnsr_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the censoring variable.

stats
(character)
the names of statistics to be reported among:

- n_tot_events: Total number of events per group.
- n_events: Number of events per group.
- n_tot: Total number of observations per group.
- n: Number of observations per group.
- median: Median survival time.
- hr: Hazard ratio.
- ci: Confidence interval of hazard ratio.
- pval: p-value of the effect. Note, one of the statistics n_tot and n_tot_events, as well as both hr and ci are required.

riskdiff
(list)
if a risk (proportion) difference column should be added, a list of settings to apply within the column. See tern::control_riskdiff() for details. If NULL, no risk difference column will be added.

conf_level (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
available confidence levels and default selection, each in the range (0, 1).

time_unit_var (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and pre-selected option for the time unit variable.

fixed_symbol_size
(logical)
When (TRUE), the same symbol size is used for plotting each estimate. Otherwise, the symbol size will be proportional to the sample size in each each subgroup.

plot_height (numeric) optional
vector of length three with c(value, min, max). Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.

plot_width (numeric) optional
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.

rel_width_forest
(proportion)
proportion of total width to allocate to the forest plot. Relative width of table is then 1 - rel_width_forest. If as_list = TRUE, this parameter is ignored.

font_size (numeric(1))
font size.

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.

ggplot2_args (ggplot2_args) optional
object created by teal.widgets::ggplot2_args() with settings for the module plot. The argument is merged with option teal.ggplot2_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-ggplot2-arguments", package = "teal.widgets").

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.

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_forest_tte(
   ..., # 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.modules.clinical")`.

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(nestcolor)

data <- teal_data()
data <- within(data, {
  library(formatters)
  library(dplyr)
  ADSL <- teal.modules.clinical::tmc_ex_adsl
  ADTTE <- teal.modules.clinical::tmc_ex_adtte
```

```
  ADSL$RACE <- droplevels(ADSL$RACE) %>% with_label("Race")
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADTTE <- data[["ADTTE"]]

arm_ref_comp <- list(
  ARM = list(
    ref = "B: Placebo",
    comp = c("A: Drug X", "C: Combination")
  ),
  ARMCD = list(
    ref = "ARM B",
    comp = c("ARM A", "ARM C")
  )
)

app <- init(
  data = data,
  modules = modules(
    tm_g_forest_tte(
      label = "Forest Survival",
      dataname = "ADTTE",
      arm_var = variables(c("ARM", "ARMCD"), "ARMCD"),
      arm_ref_comp = arm_ref_comp,
      paramcd = picks(
        variables("PARAMCD", "PARAMCD"),
        values(selected = "OS", multiple = FALSE),
        check_dataset = FALSE
      ),
      subgroup_var = variables(c("BMRKR2", "SEX"), "BMRKR2", multiple = TRUE),
      strata_var = variables(c("STRATA1", "STRATA2"), "STRATA2", multiple = TRUE)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_g_ipp *teal Module: Individual Patient Plots*

Description

This module produces `ggplot2::ggplot()` type individual patient plots that display trends in parameter values over time for each patient, using data with ADaM structure.

Usage

```
tm_g_ipp(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  paramcd,
  id_var = teal.picks::variables("USUBJID", "USUBJID", fixed = TRUE),
  visit_var = teal.picks::variables("AVISIT", "AVISIT", fixed = TRUE),
  aval_var = teal.picks::variables("AVAL", "AVAL", fixed = TRUE),
  avalu_var = teal.picks::variables("AVALU", "AVALU", fixed = TRUE),
  baseline_var = teal.picks::variables("BASE", "BASE", fixed = TRUE),
  add_baseline_hline = FALSE,
  separate_by_obs = FALSE,
  suppress_legend = FALSE,
  add_avalu = TRUE,
  plot_height = c(1200L, 400L, 5000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`, `teal.picks::picks()`, or legacy `choices_selected`)
Required (no default, as on `main`). Arm encoding; resolved on `parentname` (typically `ADSL`). Legacy inputs are coerced with a deprecation warning.

`paramcd` (`teal.picks::picks()` or legacy `choices_selected` from `teal.transform::value_choices()`)
Required (no default, as on `main`). Parameter code filter: a `teal.picks::picks()` with `teal.picks::variables()` (typically `PARAMCD`) and `teal.picks::values()` for levels; legacy `choices_selected` inputs are coerced with a deprecation warning.

`id_var` visit_var, aval_var, avalu_var, baseline_var (`teal.picks::variables()`, `teal.picks::picks()`, or legacy `choices_selected`)
encodings on `dataname`. Legacy inputs are coerced with a deprecation warning.

`visit_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)

object with all available choices and preselected option for variable names that can be used as `visit` variable. Must be a factor in `dataname`.

`aval_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

`avalu_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the analysis unit variable.

`baseline_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable values that can be used as `baseline_var`.

`add_baseline_hline`
(`logical`)
whether a horizontal line should be added to the plot at baseline y-value.

`separate_by_obs`
(`logical`)
whether to create multi-panel plots.

`suppress_legend`
(`logical`)
whether to suppress the plot legend.

`add_avalu` (`logical`)
whether `avalu_first` text should be appended to the plot title and y-axis label.

`plot_height` (`numeric`) optional
vector of length three with `c(value, min, max)`. Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.

`plot_width` (`numeric`) optional
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.

`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.

`ggplot2_args` (`ggplot2_args`) optional
object created by `teal.widgets::ggplot2_args()` with settings for the module plot. For this module, this argument will only accept `ggplot2_args` object with `labs` list of the following child elements: `title`, `subtitle`, `x`, `y`. No other elements are taken into account. The argument is merged with option `teal.ggplot2_args` and with default module arguments (hard coded in the module body).

For more details, see the vignette: `vignette("custom-ggplot2-arguments", package = "teal.widgets")`.

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.

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_ipp(
   ..., # 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.modules.clinical").

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")

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(nestcolor)

data <- teal_data()
data <- within(data, {
  library(teal.modules.clinical)
  library(tern)
  library(dplyr)
  ADSL <- tmc_ex_adsl %>%
    slice(1:20) %>%
    df_explicit_na()
  ADLB <- tmc_ex_adlb %>%
    filter(USUBJID %in% ADSL$USUBJID) %>%
    df_explicit_na() %>%
    filter(AVISIT != "SCREENING")
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADLB <- data[["ADLB"]]

app <- init(
  data = data,
  modules = modules(
    tm_g_ipp(
      label = "Individual Patient Plot",
      dataname = "ADLB",
      parentname = "ADSL",
      arm_var = variables(c("ARMCD", "ARM"), "ARMCD"),
      paramcd = picks(
        datasets("ADLB", "ADLB"),
        variables("PARAMCD", "PARAMCD"),
        values(selected = "ALT", multiple = FALSE)
      ),
      id_var = variables("USUBJID", "USUBJID"),
      visit_var = variables("AVISIT", "AVISIT"),
      aval_var = variables(c("AVAL", "CHG"), "AVAL"),
      avalu_var = variables("AVALU", "AVALU"),
      baseline_var = variables("BASE", "BASE"),
      add_baseline_hline = FALSE,
      separate_by_obs = FALSE
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

tm_g_km *teal Module: Kaplan-Meier Plot*

Description

This module produces a ggplot-style Kaplan-Meier plot for data with ADaM structure.

Usage

```
tm_g_km(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  arm_ref_comp = NULL,
  paramcd,
  strata_var,
  facet_var,
  time_unit_var = teal.picks::variables("AVALU", "AVALU", fixed = TRUE),
  aval_var = teal.picks::variables("AVAL", "AVAL", fixed = TRUE),
  cnsr_var = teal.picks::variables("CNSR", "CNSR", fixed = TRUE),
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  conf_type = teal.picks::values(c("plain", "log", "log-log"), "plain"),
  font_size = c(11L, 1L, 30),
  xticks = NULL,
  control_annot_surv_med = tern::control_surv_med_annot(),
  control_annot_coxph = tern::control_coxph_annot(x = 0.27, y = 0.35, w = 0.3),
  legend_pos = c(0.9, 0.5),
  rel_height_plot = c(80L, 0L, 100L),
  plot_height = c(800L, 400L, 5000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  transformators = list(),
  decorators = list()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.
parentname	(character) parent analysis data used in teal module, usually this refers to ADSL.

arm_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable in the results table.

arm_ref_comp
(list) optional,
if specified it must be a named list with each element corresponding to an arm variable in ADSL and the element must be another list (possibly with delayed teal.picks::variables() or teal.picks::values(); legacy teal.transform::variable_choices and teal.transform::value_choices() are deprecated but still accepted) with the elements named ref and comp that define the default reference and comparison arms when the arm variable is changed.

paramcd
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the parameter code variable from dataname.

strata_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
names of the variables for stratified analysis.

facet_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the variable used to facet the plot (from parentname). Use no selection (e.g. NULL selected) for no faceting.

time_unit_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and pre-selected option for the time unit variable.

aval_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

cnsr_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the censoring variable.

conf_level
(teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
available confidence levels and default selection, each in the range (0, 1).

conf_type
(string)
confidence interval type for median survival time CI. Options are "plain" (default), "log", "log-log".

font_size
(numeric)
numeric vector of length 3 of current, minimum and maximum font size values.

xticks
(numeric or NULL)
numeric vector of tick positions or a single number with spacing for the x-axis. If NULL (default), users can specify this interactively in the module. If provided,

the interactive input field is pre-populated with the specified values as a default. Users can then modify these values interactively, and their changes will take precedence over the default.

`control_annot_surv_med`
: (`list`)
parameters to control the position and size of the annotation table added to the plot when `annot_surv_med = TRUE`, specified using the `control_surv_med_annot()` function. Parameter options are: `x`, `y`, `w`, `h`, and `fill`. See `control_surv_med_annot()` for details.

`control_annot_coxph`
: (`list`)
parameters to control the position and size of the annotation table added to the plot when `annot_coxph = TRUE`, specified using the `control_coxph_annot()` function. Parameter options are: `x`, `y`, `w`, `h`, `fill`, and `ref_lbls`. See `control_coxph_annot()` for details.

`legend_pos`
: (`numeric(2)` or `NULL`)
vector containing x- and y-coordinates, respectively, for the legend position relative to the KM plot area. If `NULL` (default), the legend is positioned in the bottom right corner of the plot, or the middle right of the plot if needed to prevent overlapping.

`rel_height_plot`
: (`proportion`)
proportion of total figure height to allocate to the Kaplan-Meier plot. Relative height of patients at risk table is then `1 - rel_height_plot`. If `annot_at_risk = FALSE` or `as_list = TRUE`, this parameter is ignored.

`plot_height`
: (`numeric`) optional
vector of length three with `c(value, min, max)`. Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.

`plot_width`
: (`numeric`) optional
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.

`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.

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_km(
   ..., # 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.modules.clinical").

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")

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(nestcolor)

data <- teal_data()
data <- within(data, {
  library(dplyr)
  library(teal.modules.clinical)
  ADSL <- tmc_ex_adsl
  ADTTE <- tmc_ex_adtte
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADTTE <- data[["ADTTE"]]

arm_ref_comp <- list(
  ACTARMCD = list(
    ref = "ARM B",
    comp = c("ARM A", "ARM C")
  ),
  ARM = list(
    ref = "B: Placebo",
    comp = c("A: Drug X", "C: Combination")
  )
)

app <- init(
  data = data,
  modules = modules(
    tm_g_km(
      label = "Kaplan-Meier Plot",
      dataname = "ADTTE",
      parentname = "ADSL",
      arm_var = variables(
        choices = c("ARM", "ARMCD", "ACTARMCD"),
        selected = "ARM",
        multiple = FALSE
      ),
      paramcd = picks(
        datasets("ADTTE"),
        variables("PARAMCD", fixed = TRUE),
        values(
          choices = unique(ADTTE$PARAMCD),
          selected = "OS",
          multiple = FALSE
        )
      ),
      arm_ref_comp = arm_ref_comp,
      strata_var = variables(
        choices = c("SEX", "BMRKR2"),
        selected = "SEX",
        multiple = TRUE
```

```
        ),
        facet_var = variables(
          choices = c("SEX", "BMRKR2"),
          selected = NULL,
          multiple = FALSE
        ),
        xticks = c(0, 30, 60, 90, 120, 150, 180)
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

---

tm_g_lineplot *teal Module: Line Plot*

Description

This module produces a `ggplot2::ggplot()` type line plot, with optional summary table, for standard ADaM data.

Usage

```
tm_g_lineplot(
  label,
  dataname,
  parentname = "ADSL",
  strata = lifecycle::deprecated(),
  group_var = teal.picks::variables(choices = c("ARM", "ARMCD", "ACTARMCD"), selected =
    "ARM"),
  x = teal.picks::variables("AVISIT", "AVISIT"),
  y = teal.picks::variables(choices = c("AVAL", "BASE", "CHG", "PCHG"), selected =
    "AVAL"),
  y_unit = teal.picks::variables("AVALU", "AVALU"),
  paramcd = teal.picks::picks(teal.picks::variables("PARAMCD", "PARAMCD", fixed = TRUE),
   teal.picks::values(c("ALT", "CRP", "IGA"), "ALT", multiple = FALSE), check_dataset =
    FALSE),
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  interval = "mean_ci",
  mid = "mean",
  whiskers = c("mean_ci_lwr", "mean_ci_upr"),
  table = c("n", "mean_sd", "median", "range"),
  mid_type = "pl",
  mid_point_size = c(2, 1, 5),
  table_font_size = c(4, 2, 6),
  plot_height = c(1000L, 200L, 4000L),
```

```
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to ADSL.

`strata` **[Deprecated]** Please use the `group_var` argument instead.

`group_var, x, y, y_unit`
(`teal.picks::variables()`, `teal.picks::picks()`, or legacy `choices_selected`) encodings; legacy inputs are coerced with a deprecation warning.

`paramcd` (`teal.picks::picks()` or legacy `choices_selected` from `value_choices()`) parameter code filter: a `picks()` chain with the PARAMCD variable and biomarker `values()`; defaults to ALT / CRP / IGA with ALT selected.

`conf_level` (`teal.picks::values()`; legacy `teal.transform::choices_selected()` is deprecated but still accepted)
available confidence levels and default selection, each in the range (0, 1).

`interval` (character or NULL)
names of the statistics that will be plotted as intervals. All the statistics indicated in `interval` variable must be present in the object returned by `sfun`, and be of a `double` or `numeric` type vector of length two. Set `interval = NULL` if intervals should not be added to the plot.

`mid` (character or NULL)
names of the statistics that will be plotted as midpoints. All the statistics indicated in `mid` variable must be present in the object returned by `sfun`, and be of a `double` or `numeric` type vector of length one.

`whiskers` (character)
names of the interval whiskers that will be plotted. Names must match names of the list element `interval` that will be returned by `sfun` (e.g. `mean_ci_lwr` element of `sfun(x)[["mean_ci"]]`). It is possible to specify one whisker only, or to suppress all whiskers by setting `interval = NULL`.

`table` (character or NULL)
names of the statistics that will be displayed in the table below the plot. All the statistics indicated in `table` variable must be present in the object returned by `sfun`.

`mid_type` (string)
controls the type of the `mid` plot, it can be point (`"p"`), line (`"l"`), or point and line (`"pl"`).

mid_point_size (numeric(1))
font size of the mid plot points.

table_font_size
(numeric(1))
font size of the text in the table.

plot_height (numeric) optional
vector of length three with c(value, min, max). Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.

plot_width (numeric) optional
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.

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.

ggplot2_args (ggplot2_args) optional
object created by teal.widgets::ggplot2_args() with settings for the module plot. For this module, this argument will only accept ggplot2_args object with labs list of following child elements: title, subtitle, caption, y, lty. No other elements would be taken into account. The argument is merged with option teal.ggplot2_args and with default module arguments (hard coded in the module body).

For more details, see the vignette: vignette("custom-ggplot2-arguments", package = "teal.widgets").

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.

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_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.modules.clinical")`.

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(nestcolor)

data <- teal_data()
data <- within(data, {
  library(teal.modules.clinical)
  library(dplyr)
  library(forcats)
  ADSL <- tmc_ex_adsl
  ADLB <- tmc_ex_adlb %>%
    mutate(AVISIT == fct_reorder(AVISIT, AVISITN, min))
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_g_lineplot(
      label = "Line Plot",
```

```
        dataname = "ADLB",
        parentname = "ADSL",
        group_var = variables(
          choices = c("ARM", "ARMCD", "ACTARMCD"),
          selected = "ARM",
          multiple = FALSE
        ),
        y = variables(
          choices = c("AVAL", "BASE", "CHG", "PCHG"),
          selected = "AVAL",
          multiple = FALSE
        ),
        paramcd = picks(
          variables("PARAMCD", fixed = TRUE),
          values(selected = "ALT", multiple = FALSE),
          check_dataset = FALSE
        )
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

---

tm_g_pp_adverse_events

teal Module: Patient Profile Adverse Events Table and Plot

Description

This module produces an adverse events table and ggplot2::ggplot() type plot using ADaM datasets.

Usage

```
tm_g_pp_adverse_events(
  label,
  dataname = "ADAE",
  parentname = "ADSL",
  patient_col = "USUBJID",
  aeterm = NULL,
  tox_grade = NULL,
  causality = NULL,
  outcome = NULL,
  action = NULL,
  time = NULL,
  decod = NULL,
  font_size = c(12L, 12L, 25L),
```

```
  plot_height = c(700L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.
parentname	(character) parent analysis data used in teal module, usually this refers to ADSL.
patient_col	(character) name of patient ID variable.
aeterm	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the AETERM variable from dataname.
tox_grade	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the AETOXGR variable from dataname.
causality	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the AEREL variable from dataname.
outcome	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the AEOUT variable from dataname.
action	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the AEACN variable from dataname.
time	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the ASTDY variable from dataname.
decod	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the AEDECOD variable from dataname.
font_size	(numeric) numeric vector of length 3 of current, minimum and maximum font size values.
plot_height	(numeric) optional vector of length three with c(value, min, max). Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.

plot_width (numeric) optional
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.

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.

ggplot2_args (ggplot2_args) optional
object created by teal.widgets::ggplot2_args() with settings for the module plot. The argument is merged with option teal.ggplot2_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-ggplot2-arguments", package = "teal.widgets").

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.

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_pp_adverse_events(
   ..., # 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.modules.clinical").

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")`

Examples in Shinylive

example-1 Open in Shinylive

Examples

```
library(nestcolor)

data <- teal_data()
data <- within(data, {
  ADAE <- teal.modules.clinical::tmc_ex_adae
  ADSL <- dplyr::filter(teal.modules.clinical::tmc_ex_adsl, USUBJID %in% ADAE$USUBJID)
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADAE <- data[["ADAE"]]

app <- init(
  data = data,
  modules = modules(
    tm_g_pp_adverse_events(
      label = "Adverse Events",
      dataname = "ADAE",
      parentname = "ADSL",
      patient_col = "USUBJID",
      plot_height = c(600L, 200L, 2000L),
      aeterm = variables("AETERM"),
      tox_grade = variables("AETOXGR"),
      causality = variables("AEREL"),
      outcome = variables("AEOUT"),
      action = variables("AEACN"),
      time = variables("ASTDY"),
      decod = NULL
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

tm_g_pp_patient_timeline

teal Module: Patient Profile Timeline Plot

Description

This module produces a patient profile timeline `ggplot2::ggplot()` type plot using ADaM datasets.

Usage

```
tm_g_pp_patient_timeline(
  label,
  dataname_adcm = "ADCM",
  dataname_adae = "ADAE",
  parentname = "ADSL",
  patient_col = "USUBJID",
  aeterm = NULL,
  cmdecod = NULL,
  aetime_start = NULL,
  aetime_end = NULL,
  dstime_start = NULL,
  dstime_end = NULL,
  aerelday_start = NULL,
  aerelday_end = NULL,
  dsrelday_start = NULL,
  dsrelday_end = NULL,
  font_size = c(12L, 12L, 25L),
  plot_height = c(700L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label`	`(character)` menu item label of the module in the teal app.
`dataname_adcm`	`(character)` name of `ADCM` dataset or equivalent.
`dataname_adae`	`(character)` name of `ADAE` dataset or equivalent.
`parentname`	`(character)` parent analysis data used in teal module, usually this refers to `ADSL`.

patient_col (character)
name of patient ID variable.

aeterm (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the AETERM variable from dataname_adae.

cmdecod (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMDECOD variable from dataname_adcm.

aetime_start (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the ASTDTM variable from dataname_adae.

aetime_end (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the AENDTM variable from dataname_adae.

dstime_start (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMASTDTM variable from dataname_adcm.

dstime_end (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMAENDTM variable from dataname_adcm.

aerelday_start (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the ASTDY variable from dataname_adae.

aerelday_end (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the AENDY variable from dataname_adae.

dsrelday_start (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the ASTDY variable from dataname_adcm.

dsrelday_end (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the AENDY variable from dataname_adcm.

font_size (numeric)
numeric vector of length 3 of current, minimum and maximum font size values.

plot_height (numeric) optional
vector of length three with c(value, min, max). Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.

plot_width (numeric) optional
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.

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.

ggplot2_args (ggplot2_args) optional
object created by teal.widgets::ggplot2_args() with settings for the module plot. The argument is merged with option teal.ggplot2_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-ggplot2-arguments", package = "teal.widgets").

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.

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_pp_patient_timeline(
   ..., # 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.modules.clinical")`.

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")`

Examples in Shinylive

example-1 Open in Shinylive

Examples

```
library(nestcolor)

data <- teal_data()
data <- within(data, {
  library(teal.modules.clinical)
  library(formatters)
  library(dplyr)
  ADAE <- tmc_ex_adae
  ADSL <- filter(tmc_ex_adsl, USUBJID %in% ADAE$USUBJID)
  ADCM <- tmc_ex_adcm %>%
    mutate(
      CMSTDY = case_when(
        CMCAT == "medcl B" ~ 20,
        CMCAT == "medcl C" ~ 150,
        TRUE ~ 1
      ) %>% with_label("Study Day of Start of Medication"),
      CMENDY = case_when(
        CMCAT == "medcl B" ~ 700,
        CMCAT == "medcl C" ~ 1000,
        TRUE ~ 500
      ) %>% with_label("Study Day of End of Medication"),
      CMASTDTM = ASTDTM,
      CMAENDTM = AENDTM
    )
})

join_keys(data) <- default_cdisc_join_keys[c("ADSL", "ADAE", "ADCM")]
adcm_keys <- c("STUDYID", "USUBJID", "ASTDTM", "CMSEQ", "ATC1", "ATC2", "ATC3", "ATC4")
join_keys(data)["ADCM", "ADCM"] <- adcm_keys
join_keys(data)["ADAE", "ADCM"] <- c("STUDYID", "USUBJID")
```

```
app <- init(
  data = data,
  modules = modules(
    tm_g_pp_patient_timeline(
      label = "Patient Timeline",
      dataname_adae = "ADAE",
      dataname_adcm = "ADCM",
      parentname = "ADSL",
      patient_col = "USUBJID",
      plot_height = c(600L, 200L, 2000L),
      cmdecod = variables("CMDECOD", "CMDECOD"),
      aeterm = variables("AETERM", "AETERM"),
      aetime_start = variables("ASTDTM", "ASTDTM"),
      aetime_end = variables("AENDTM", "AENDTM"),
      dstime_start = variables("CMASTDTM", "CMASTDTM"),
      dstime_end = variables("CMAENDTM", "CMAENDTM"),
      aerelday_start = variables("ASTDY", "ASTDY"),
      aerelday_end = variables("AENDY", "AENDY"),
      dsrelday_start = variables("ASTDY", "ASTDY"),
      dsrelday_end = variables("AENDY", "AENDY")
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_g_pp_therapy *teal Module: Patient Profile Therapy Table and Plot*

Description

This module produces a patient profile therapy table and `ggplot2::ggplot()` type plot using ADaM datasets.

Usage

```
tm_g_pp_therapy(
  label,
  dataname = "ADCM",
  parentname = "ADSL",
  patient_col = "USUBJID",
  atirel = NULL,
  cmdecod = NULL,
  cmindc = NULL,
  cmdose = NULL,
  cmtrt = NULL,
```

```
  cmdosu = NULL,
  cmroute = NULL,
  cmdosfrq = NULL,
  cmstdy = NULL,
  cmendy = NULL,
  font_size = c(12L, 12L, 25L),
  plot_height = c(700L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`patient_col` (character)
name of patient ID variable.

`atirel` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the `ATIREL` variable from `dataname`.

`cmdecod` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the `CMDECOD` variable from `dataname`.

`cmindc` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the `CMINDC` variable from `dataname`.

`cmdose` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the `CMDOSE` variable from `dataname`.

`cmtrt` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the `CMTRT` variable from `dataname`.

cmdosu
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMDOSU variable from dataname.

cmroute
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMROUTE variable from dataname.

cmdosfrq
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMDOSFRQ variable from dataname.

cmstdy
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMSTDY variable from dataname.

cmendy
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMENDY variable from dataname.

font_size
: (numeric)
numeric vector of length 3 of current, minimum and maximum font size values.

plot_height
: (numeric) optional
vector of length three with c(value, min, max). Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.

plot_width
: (numeric) optional
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.

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.

ggplot2_args
: (ggplot2_args) optional
object created by teal.widgets::ggplot2_args() with settings for the module plot. The argument is merged with option teal.ggplot2_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-ggplot2-arguments", package = "teal.widgets").

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.

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_pp_therapy(
   ..., # 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.modules.clinical")`.

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

Examples in Shinylive

example-1 Open in Shinylive

Examples

```
library(nestcolor)
library(dplyr)

data <- teal_data()
data <- within(data, {
  ADCM <- tmc_ex_adcm
  ADSL <- tmc_ex_adsl %>% filter(USUBJID %in% ADCM$USUBJID)
  ADCM$CMASTDTM <- ADCM$ASTDTM
  ADCM$CMAENDTM <- ADCM$AENDTM
})

join_keys(data) <- default_cdisc_join_keys[c("ADSL", "ADCM")]
adcm_keys <- c("STUDYID", "USUBJID", "ASTDTM", "CMSEQ", "ATC1", "ATC2", "ATC3", "ATC4")
join_keys(data)["ADCM", "ADCM"] <- adcm_keys

ADSL <- data[["ADSL"]]
ADCM <- data[["ADCM"]]

app <- init(
```

```
    data = data,
    modules = modules(
      tm_g_pp_therapy(
        label = "Therapy",
        dataname = "ADCM",
        parentname = "ADSL",
        patient_col = "USUBJID",
        plot_height = c(600L, 200L, 2000L),
        atirel = variables(choices = "ATIREL", selected = "ATIREL"),
        cmdecod = variables(choices = "CMDECOD", selected = "CMDECOD"),
        cmindc = variables(choices = "CMINDC", selected = "CMINDC"),
        cmdose = variables(choices = "CMDOSE", selected = "CMDOSE"),
        cmtrt = variables(choices = "CMTRT", selected = "CMTRT"),
        cmdosu = variables(choices = "CMDOSU", selected = "CMDOSU"),
        cmroute = variables(choices = "CMROUTE", selected = "CMROUTE"),
        cmdosfrq = variables(choices = "CMDOSFRQ", selected = "CMDOSFRQ"),
        cmstdy = variables(choices = "ASTDY", selected = "ASTDY"),
        cmendy = variables(choices = "AENDY", selected = "AENDY")
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

`tm_g_pp_vitals` *teal Module: Patient Profile Vitals Plot*

Description

This module produces a patient profile vitals `ggplot2::ggplot()` type plot using ADaM datasets.

Usage

```
tm_g_pp_vitals(
  label,
  dataname = "ADVS",
  parentname = "ADSL",
  patient_col = "USUBJID",
  paramcd = NULL,
  aval = lifecycle::deprecated(),
  aval_var = NULL,
  xaxis = NULL,
  font_size = c(12L, 12L, 25L),
  plot_height = c(700L, 200L, 2000L),
  plot_width = NULL,
  pre_output = NULL,
  post_output = NULL,
```

```
  ggplot2_args = teal.widgets::ggplot2_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.
parentname	(character) parent analysis data used in teal module, usually this refers to ADSL.
patient_col	(character) name of patient ID variable.
paramcd	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for the parameter code variable from dataname.
aval	**[Deprecated]** Please use the aval_var argument instead.
aval_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and pre-selected option for the analysis variable.
xaxis	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the time variable from dataname to be put on the plot x-axis.
font_size	(numeric) numeric vector of length 3 of current, minimum and maximum font size values.
plot_height	(numeric) optional vector of length three with c(value, min, max). Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.
plot_width	(numeric) optional 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.
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.
ggplot2_args	(ggplot2_args) optional object created by teal.widgets::ggplot2_args() with settings for the module plot. The argument is merged with option teal.ggplot2_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-ggplot2-arguments", package = "teal.widgets").

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.

Details

This plot supports horizontal lines for the following 6 PARAMCD levels when they are present in dataname: "SYSBP", "DIABP", "TEMP", "RESP", "OXYSAT".

Value

a teal_module object.

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_pp_vitals(
   ..., # 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.modules.clinical").

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")

Examples in Shinylive

example-1 Open in Shinylive

Examples

```
library(nestcolor)

data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADVS <- tmc_ex_advs
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADVS <- data[["ADVS"]]

app <- init(
  data = data,
  modules = modules(
    tm_g_pp_vitals(
      label = "Vitals",
      dataname = "ADVS",
      parentname = "ADSL",
      patient_col = "USUBJID",
      plot_height = c(600L, 200L, 2000L),
      paramcd = variables("PARAMCD"),
      aval_var = variables("AVAL"),
      xaxis = variables("ADY"),
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_t_abnormality *teal Module: Abnormality Summary Table*

Description

This module produces a table to summarize abnormality.

Usage

```
tm_t_abnormality(
  label,
  dataname,
```

```
  parentname = "ADSL",
  arm_var,
  by_vars,
  grade,
  abnormal = list(low = c("LOW", "LOW LOW"), high = c("HIGH", "HIGH HIGH")),
  id_var = teal.picks::variables("USUBJID", "USUBJID", fixed = TRUE),
  baseline_var = teal.picks::variables("BNRIND", "BNRIND", fixed = TRUE),
  treatment_flag_var = teal.picks::variables("ONTRTFL", "ONTRTFL", fixed = TRUE),
  treatment_flag = teal.picks::values("Y", "Y", fixed = TRUE, multiple = FALSE),
  add_total = TRUE,
  total_label = default_total_label(),
  exclude_base_abn = FALSE,
  drop_arm_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
  na_level = tern::default_na_str(),
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `arm_var`. It defines the grouping variable in the results table.

`by_vars` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option(s) for row-by variables (`multiple = TRUE`, `ordered = TRUE` recommended).

`grade` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the abnormality grade variable. Variable must be factor.

`abnormal` (named list)
defined by user to indicate what abnormalities are to be displayed.

`id_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object specifying the variable name for subject id.

baseline_var (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
variable for baseline abnormality grade.

treatment_flag_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
on treatment flag variable.

treatment_flag (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
value(s) indicating on-treatment records in treatment_flag_var (e.g. "Y").

add_total (logical)
whether to include column with total number of patients.

total_label (string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

exclude_base_abn
(logical)
whether to exclude patients who had abnormal values at baseline.

drop_arm_levels
(logical)
whether to drop unused levels of arm_var. If TRUE, arm_var levels are set to those used in the dataname dataset. If FALSE, arm_var levels are set to those used in the parentname dataset. If dataname and parentname are the same, then drop_arm_levels is set to TRUE and user input for this parameter is ignored.

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.

na_level (character)
the NA level in the input dataset, defaults to "<Missing>".

basic_table_args
(basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- `table` (`TableTree` - output of `rtables::build_table()`)

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

Note

Patients with the same abnormality at baseline as on the treatment visit can be excluded in accordance with GDSR specifications by using `exclude_base_abn`.

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  library(teal.modules.clinical)
  library(formatters)
  library(dplyr)
  ADSL <- tmc_ex_adsl
  ADLB <- tmc_ex_adlb %>%
    mutate(
      ONTRTFL = case_when(
        AVISIT %in% c("SCREENING", "BASELINE") ~ "",
        TRUE ~ "Y"
      ) %>% with_label("On Treatment Record Flag")
    )
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADLB <- data[["ADLB"]]

app <- init(
  data = data,
  modules = modules(
    tm_t_abnormality(
      label = "Abnormality Table",
      dataname = "ADLB",
      arm_var = variables(choices = c("ARM", "ARMCD"), selected = "ARM"),
      add_total = FALSE,
      by_vars = variables(
        choices = c("LBCAT", "PARAM", "AVISIT"),
        selected = c("LBCAT", "PARAM"),
        multiple = TRUE,
        ordered = TRUE
      ),
      baseline_var = variables(choices = "BNRIND", fixed = TRUE),
      grade = variables(choices = "ANRIND", fixed = TRUE),
      abnormal = list(low = "LOW", high = "HIGH"),
      exclude_base_abn = FALSE
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

tm_t_abnormality_by_worst_grade

teal Module: Laboratory test results with highest grade post-baseline

Description

This module produces a table to summarize laboratory test results with highest grade post-baseline

Usage

```
tm_t_abnormality_by_worst_grade(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  id_var = teal.picks::variables("USUBJID", "USUBJID", fixed = TRUE),
  paramcd,
  atoxgr_var = teal.picks::variables("ATOXGR", "ATOXGR", fixed = TRUE),
 worst_high_flag_var = teal.picks::variables("WGRHIFL", "WGRHIFL", fixed = TRUE),
 worst_low_flag_var = teal.picks::variables("WGRLOFL", "WGRLOFL", fixed = TRUE),
 worst_flag_indicator = teal.picks::values("Y", "Y", fixed = FALSE, multiple = FALSE),
  add_total = TRUE,
  total_label = default_total_label(),
  drop_arm_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label`	(character) menu item label of the module in the teal app.
`dataname`	(character) analysis data used in teal module.
`parentname`	(character) parent analysis data used in teal module, usually this refers to `ADSL`.
`arm_var`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) object with all available choices and preselected option for variable names that can be used as `arm_var`. It defines the grouping variable in the results table.
`id_var`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) object specifying the variable name for subject id.

paramcd
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
variable used to filter the analysis dataset (typically PARAMCD). The values() element is added internally to allow users to pick laboratory parameter value(s) interactively.

atoxgr_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the analysis toxicity grade variable.

worst_high_flag_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the worst high grade flag variable.

worst_low_flag_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the worst low grade flag variable.

worst_flag_indicator
: (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
Value(s) matching the worst high/low flag variables (default "Y"). Uses explicit candidate levels including an empty string where needed. The UI shows the selected value as static text (not an interactive control).

add_total
: (logical)
whether to include column with total number of patients.

total_label
: (string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

drop_arm_levels
: (logical)
whether to drop unused levels of arm_var. If TRUE, arm_var levels are set to those used in the dataname dataset. If FALSE, arm_var levels are set to those used in the parentname dataset. If dataname and parentname are the same, then drop_arm_levels is set to TRUE and user input for this parameter is ignored.

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.

basic_table_args
: (basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the

module table. The argument is merged with option `teal.basic_table_args` and with default module arguments (hard coded in the module body). For more details, see the vignette: `vignette("custom-basic-table-arguments", package = "teal.widgets")`.

`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.

Decorating Module

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

- `table` (`TableTree` - output of `rtables::build_table()`)

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(dplyr)

data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADLB <- tmc_ex_adlb %>%
    filter(!AVISIT %in% c("SCREENING", "BASELINE"))
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADLB <- data[["ADLB"]]

app <- init(
  data = data,
  modules = modules(
    tm_t_abnormality_by_worst_grade(
      label = "Laboratory Test Results with Highest Grade Post-Baseline",
      dataname = "ADLB",
      arm_var = variables(choices = c("ARM", "ARMCD"), selected = "ARM"),
      paramcd = variables(choices = "PARAMCD"),
      add_total = FALSE
    )
  ),
  filter = teal_slices(
    teal_slice("ADSL", "SAFFL", selected = "Y"),
    teal_slice("ADLB", "ONTRTFL", selected = "Y")
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_t_ancova *teal Module: ANCOVA Summary*

Description

This module produces a table to summarize analysis of variance, consistent with the TLG Catalog template for `AOVT01` available here when multiple endpoints are selected.

Usage

```
tm_t_ancova(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  arm_ref_comp = NULL,
  aval_var,
  cov_var,
  include_interact = FALSE,
  interact_var = NULL,
  interact_y = FALSE,
  avisit,
  paramcd,
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `arm_var`. It defines the grouping variable in the results table.

`arm_ref_comp` (`list`) optional,
if specified it must be a named list with each element corresponding to an arm variable in `ADSL` and the element must be another list (possibly with delayed `teal.picks::variables()` or `teal.picks::values()`; legacy `teal.transform::variable_choices` and `teal.transform::value_choices()` are deprecated but still accepted) with the elements named `ref` and `comp` that define the default reference and comparison arms when the arm variable is changed.

`aval_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

`cov_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)

object with all available choices and preselected option for the covariates variables.

include_interact
: (logical)
whether an interaction term should be included in the model.

interact_var
: (character)
name of the variable that should have interactions with arm. If the interaction is not needed, the default option is NULL.

interact_y
: (character)
a selected item from the interact_var column which will be used to select the specific ANCOVA results when interact_var is discrete. If the interaction is not needed, the default option is FALSE.

avisit
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
value of analysis visit AVISIT of interest.

paramcd
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the parameter code variable from dataname.

conf_level
: (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
available confidence levels and default selection, each in the range (0, 1).

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.

basic_table_args
: (basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Details

When a single endpoint is selected, both unadjusted and adjusted comparison are provided. This modules expects that the analysis data has the following variables:

- AVISIT: variable used to filter for analysis visits.
- PARAMCD: variable used to filter for endpoints, after filtering for paramcd and avisit, one observation per patient is expected for the analysis to be meaningful.

Value

a teal_module object.

Decorating Module

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

- table (TableTree - output of rtables::build_table())

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

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

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")

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADQS <- tmc_ex_adqs
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADQS <- data[["ADQS"]]

arm_ref_comp <- list(
  ARM = list(
    ref = "B: Placebo",
    comp = c("A: Drug X", "C: Combination")
  ),
  ACTARMCD = list(
    ref = "ARM B",
    comp = c("ARM A", "ARM C")
  )
)

app <- init(
  data = data,
  modules = modules(
    tm_t_ancova(
      label = "ANCOVA Table",
      dataname = "ADQS",
      avisit = picks(
        variables("AVISIT", "AVISIT"),
        values(selected = "WEEK 1 DAY 8", multiple = TRUE),
        check_dataset = FALSE
      ),
      arm_var = variables(c("ARM", "ACTARMCD", "ARMCD"), selected = "ARMCD"),
      arm_ref_comp = arm_ref_comp,
      aval_var = variables(c("CHG", "AVAL"), selected = "CHG", multiple = FALSE),
      cov_var = variables(c("BASE", "STRATA1", "SEX"), selected = "STRATA1"),
      paramcd = picks(
        variables("PARAMCD", "PARAMCD"),
        values(selected = "FKSI-FWB", multiple = TRUE),
        check_dataset = FALSE
      ),
    interact_var = variables(c("BASE", "STRATA1", "SEX"), selected = "STRATA1", multiple = FALSE)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

tm_t_binary_outcome *teal Module: Binary Outcome Table*

Description

This module produces a binary outcome response summary table, with the option to match the template for response table RSPT01 available in the TLG Catalog here.

Usage

```
tm_t_binary_outcome(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  arm_ref_comp = NULL,
  paramcd,
  strata_var,
  aval_var = teal.picks::variables(c("AVALC", "SEX"), "AVALC", fixed = FALSE),
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  default_responses = c("CR", "PR", "Y", "Complete Response (CR)",
    "Partial Response (PR)", "M"),
  rsp_table = FALSE,
 control = list(global = list(method = ifelse(rsp_table, "clopper-pearson", "waldcc"),
   conf_level = 0.95), unstrat = list(method_ci = ifelse(rsp_table, "wald", "waldcc"),
   method_test = "schouten", odds = TRUE), strat = list(method_ci = "cmh", method_test =
    "cmh")),
  add_total = FALSE,
  total_label = default_total_label(),
  na_level = tern::default_na_str(),
  denom = c("N_col", "n", "N_row"),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.
parentname	(character) parent analysis data used in teal module, usually this refers to ADSL.

arm_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable in the results table.

arm_ref_comp
: (list) optional,
if specified it must be a named list with each element corresponding to an arm variable in ADSL and the element must be another list (possibly with delayed teal.picks::variables() or teal.picks::values(); legacy teal.transform::variable_choices and teal.transform::value_choices() are deprecated but still accepted) with the elements named ref and comp that define the default reference and comparison arms when the arm variable is changed.

paramcd
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the parameter code variable from dataname.

strata_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
names of the variables for stratified analysis.

aval_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

conf_level
: (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
available confidence levels and default selection in (0, 1). Choice order follows the vector passed to teal.picks::values() (there is no keep_order argument).

default_responses
: (list or character)
defines the default codes for the response variable in the module per value of paramcd. A passed vector is transmitted for all paramcd values. A passed list must be named and contain arrays, each name corresponding to a single value of paramcd. Each array may contain default response values or named arrays rsp of default selected response values and levels of default level choices.

rsp_table
: (logical)
whether the initial set-up of the module should match RSPT01. Defaults to FALSE.

control
: (named list)
named list containing 3 named lists as follows:

- global: a list of settings for overall analysis with 2 named elements method and conf_level.
- unstrat: a list of settings for unstratified analysis with 3 named elements method_ci and method_test, and odds. See tern::estimate_proportion_diff(), tern::test_proportion_diff(), and tern::estimate_odds_ratio(), respectively, for options and details on how these settings are implemented in the analysis.

- strat: a list of settings for stratified analysis with elements method_ci and method_test. See tern::estimate_proportion_diff() and tern::test_proportion_diff(), respectively, for options and details on how these settings are implemented in the analysis.

add_total (logical)
whether to include column with total number of patients.

total_label (string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

na_level (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

denom (string)
choice of denominator for proportion. Options are:

- N_col: total number of patients in this column across rows.
- n: number of patients with any occurrences.
- N_row: total number of patients in this row across columns.

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.

basic_table_args
(basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Details

- The display order of response categories inherits the factor level order of the source data. Use base::factor() and its levels argument to manipulate the source data in order to include/exclude or re-categorize response categories and arrange the display order. If response

categories are "Missing", "Not Evaluable (NE)", or "Missing or unevaluable", 95% confidence interval will not be calculated.

- Reference arms are automatically combined if multiple arms selected as reference group.

Value

a teal_module object.

Decorating Module

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

- table (TableTree - output of rtables::build_table())

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

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

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")

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  library(dplyr)
  library(formatters)
  ADSL <- tmc_ex_adsl
  ADRS <- tmc_ex_adrs %>%
    mutate(
      AVALC = d_onco_rsp_label(AVALC) %>%
        with_label("Character Result/Finding")
    ) %>%
    filter(PARAMCD != "OVRINV" | AVISIT == "FOLLOW UP")
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADRS <- data[["ADRS"]]

arm_ref_comp <- list(
  ARMCD = list(ref = "ARM B", comp = c("ARM A", "ARM C")),
  ARM = list(ref = "B: Placebo", comp = c("A: Drug X", "C: Combination"))
)
app <- init(
  data = data,
  modules = modules(
    tm_t_binary_outcome(
      label = "Responders",
      dataname = "ADRS",
      paramcd = variables(choices = "PARAMCD"),
      arm_var = variables(
        choices = c("ARM", "ARMCD", "ACTARMCD"),
        selected = "ARM"
      ),
      arm_ref_comp = arm_ref_comp,
      strata_var = variables(
        choices = c("SEX", "BMRKR2", "RACE"),
        selected = "RACE"
      ),
      default_responses = list(
        BESRSPI = list(
          rsp = c("Complete Response (CR)", "Partial Response (PR)"),
          levels = c(
            "Complete Response (CR)", "Partial Response (PR)",
            "Stable Disease (SD)", "Progressive Disease (PD)"
          )
        ),
        INVET = list(
          rsp = c("Stable Disease (SD)", "Not Evaluable (NE)"),
          levels = c(
            "Complete Response (CR)", "Not Evaluable (NE)", "Partial Response (PR)",
            "Progressive Disease (PD)", "Stable Disease (SD)"
          )
```

```
          ),
          OVRINV = list(
            rsp = c("Progressive Disease (PD)", "Stable Disease (SD)"),
          levels = c("Progressive Disease (PD)", "Stable Disease (SD)", "Not Evaluable (NE)")
          )
        ),
        denom = "N_col"
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

---

tm_t_coxreg *teal Module: Cox Regression Model*

Description

This module fits Cox univariable or multi-variable models, consistent with the TLG Catalog templates for Cox regression tables COXT01 and COXT02, respectively. See the TLG Catalog entries for COXT01 here and COXT02 here.

Usage

```
tm_t_coxreg(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  arm_ref_comp = NULL,
  paramcd,
  cov_var,
  strata_var,
  aval_var = teal.picks::variables(choices = "AVAL", "AVAL", fixed = TRUE),
  cnsr_var = teal.picks::variables(choices = "CNSR", "CNSR", fixed = TRUE),
  multivariate = TRUE,
  na_level = tern::default_na_str(),
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.
parentname	(character) parent analysis data used in teal module, usually this refers to ADSL.
arm_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable in the results table.
arm_ref_comp	(list) optional, if specified it must be a named list with each element corresponding to an arm variable in ADSL and the element must be another list (possibly with delayed teal.picks::variables() or teal.picks::values(); legacy teal.transform::variable_choices and teal.transform::value_choices() are deprecated but still accepted) with the elements named ref and comp that define the default reference and comparison arms when the arm variable is changed.
paramcd	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for the parameter code variable from dataname.
cov_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for the covariates variables.
strata_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) names of the variables for stratified analysis.
aval_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and pre-selected option for the analysis variable.
cnsr_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for the censoring variable.
multivariate	(logical) if FALSE, the univariable approach is used instead of the multi-variable model.
na_level	(string) used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").
conf_level	(teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)

available confidence levels and default selection, each in the range (0, 1). Choice order follows the vector passed to `teal.picks::values()` (there is no `keep_order` argument).

`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.

`basic_table_args`
(`basic_table_args`) optional
object created by `teal.widgets::basic_table_args()` with settings for the module table. The argument is merged with option `teal.basic_table_args` and with default module arguments (hard coded in the module body). For more details, see the vignette: `vignette("custom-basic-table-arguments", package = "teal.widgets")`.

`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.

Details

The Cox Proportional Hazards (PH) model is the most commonly used method to estimate the magnitude of the effect in survival analysis. It assumes proportional hazards: the ratio of the hazards between groups (e.g., two arms) is constant over time. This ratio is referred to as the "hazard ratio" (HR) and is one of the most commonly reported metrics to describe the effect size in survival analysis.

This modules expects that the analysis data has the following variables:

- `AVAL`: time to event
- `CNSR`: 1 if record in `AVAL` is censored, 0 otherwise
- `PARAMCD`: variable used to filter for endpoint (e.g. OS). After filtering for `PARAMCD` one observation per patient is expected

The arm variables and stratification/covariate variables are taken from the `ADSL` data.

Value

a `teal_module` object.

Decorating Module

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

- `table` (`TableTree` as created from `rtables::build_table`)

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

example-2 Open in Shinylive

Note

- The likelihood ratio test is not supported for models that include strata - the Wald test will be substituted in these cases.
- Multi-variable is the default choice for backward compatibility.

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
## First example
## =============
## The example below is based on the usual approach involving creation of
## a random CDISC dataset and then running the application.

arm_ref_comp <- list(
  ACTARMCD = list(
    ref = "ARM B",
    comp = c("ARM A", "ARM C")
  ),
  ARM = list(
    ref = "B: Placebo",
    comp = c("A: Drug X", "C: Combination")
  )
)

data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADTTE <- tmc_ex_adtte
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADTTE <- data[["ADTTE"]]

app <- init(
  data = data,
  modules = modules(
    tm_t_coxreg(
      label = "Cox Reg.",
      dataname = "ADTTE",
      arm_var = variables(
        choices = c("ARM", "ARMCD", "ACTARMCD"),
        selected = "ARM"
      ),
      arm_ref_comp = arm_ref_comp,
      paramcd = variables(choices = "PARAMCD"),
      strata_var = variables(
        choices = c("COUNTRY", "STRATA1", "STRATA2"),
        selected = "STRATA1"
      ),
      cov_var = variables(
        choices = c("AGE", "BMRKR1", "BMRKR2", "REGION1"),
        selected = "AGE",
        multiple = TRUE,
        ordered = TRUE
      ),
      multivariate = TRUE
    )
  )
```

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

## Second example
## ==============
## This time, a synthetic pair of ADTTE/ADSL data is fabricated for Cox regression
## where ties and pval_method matter.
library(dplyr)

data <- teal_data()
data <- within(data, {
  ADTTE <- data.frame(
    STUDYID = "LUNG",
    AVAL = c(4, 3, 1, 1, 2, 2, 3, 1, 2),
    CNSR = c(1, 1, 1, 0, 1, 1, 0, 0, 0),
    ARMCD = factor(
      c(0, 1, 1, 1, 1, 0, 0, 0, 0),
      labels = c("ARM A", "ARM B")
    ),
    SEX = factor(
      c(0, 0, 0, 0, 1, 1, 1, 1, 1),
      labels = c("F", "M")
    ),
    INST = factor(c("A", "A", "B", "B", "A", "B", "A", "B", "A")),
    stringsAsFactors = FALSE
  )
  ADTTE <- rbind(ADTTE, ADTTE, ADTTE, ADTTE)
  ADTTE <- as_tibble(ADTTE)
  set.seed(1)
  ADTTE$INST <- sample(ADTTE$INST)
  ADTTE$AGE <- sample(seq(5, 75, 5), size = nrow(ADTTE), replace = TRUE)
  ADTTE$USUBJID <- paste("sub", 1:nrow(ADTTE), ADTTE$INST, sep = "-")
  ADTTE$PARAM <- ADTTE$PARAMCD <- "OS"
  ADSL <- subset(
    ADTTE,
    select = c("USUBJID", "STUDYID", "ARMCD", "SEX", "INST", "AGE")
  )
})

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

ADSL <- data[["ADSL"]]
ADTTE <- data[["ADTTE"]]

## `teal` application
## ----------------
## Note that the R code exported by `Show R Code` does not include the data
## pre-processing. You will need to create the dataset as above before
## running the exported R code.

arm_ref_comp <- list(ARMCD = list(ref = "ARM A", comp = c("ARM B")))
```

```
app <- init(
  data = data,
  modules = modules(
    tm_t_coxreg(
      label = "Cox Reg.",
      dataname = "ADTTE",
      arm_var = variables(choices = "ARMCD"),
      arm_ref_comp = arm_ref_comp,
      paramcd = variables(choices = "PARAMCD"),
      strata_var = variables(choices = "INST", selected = NULL),
      cov_var = variables(
        choices = c("SEX", "AGE"),
        selected = "SEX",
        multiple = TRUE,
        ordered = TRUE
      ),
      multivariate = TRUE
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_t_events *teal Module: Events by Term*

Description

This module produces a table of events by term.

Usage

```
tm_t_events(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  hlt,
  llt,
  add_total = TRUE,
  total_label = default_total_label(),
  na_level = tern::default_na_str(),
  event_type = "event",
  sort_criteria = c("freq_desc", "alpha"),
  sort_freq_col = total_label,
  prune_freq = 0,
```

```
  prune_diff = 0,
  drop_arm_levels = TRUE,
  incl_overall_sum = TRUE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
If there are two elements selected for `arm_var`, the second variable is nested under the first.

`hlt` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
name of the variable with high level term for events.

`llt` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
name of the variable with low level term for events.

`add_total` (logical)
whether to include column with total number of patients.

`total_label` (string)
string to display as total column/row label if column/row is enabled (see `add_total`). Defaults to `"All Patients"`. To set a new default `total_label` to apply in all modules, run `set_default_total_label("new_default")`.

`na_level` (string)
used to replace all `NA` or empty values in character or factor variables in the data. Defaults to `"<Missing>"`. To set a default `na_level` to apply in all modules, run `set_default_na_str("new_default")`.

`event_type` (character)
type of event that is summarized (e.g. adverse event, treatment). Default is `"event"`.

`sort_criteria` (character)
how to sort the final table. Default option `freq_desc` sorts on column `sort_freq_col` by decreasing number of patients with event. Alternative option `alpha` sorts events alphabetically.

sort_freq_col (character)
column to sort by frequency on if sort_criteria is set to freq_desc.

prune_freq (number)
threshold to use for trimming table using event incidence rate in any column.

prune_diff (number)
threshold to use for trimming table using as criteria difference in rates between any two columns.

drop_arm_levels
(logical)
whether to drop unused levels of arm_var. If TRUE, arm_var levels are set to those used in the dataname dataset. If FALSE, arm_var levels are set to those used in the parentname dataset. If dataname and parentname are the same, then drop_arm_levels is set to TRUE and user input for this parameter is ignored.

incl_overall_sum
(flag)
whether two rows which summarize the overall number of adverse events should be included at the top of the table.

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.

basic_table_args
(basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- table (TableTree as created from rtables::build_table)

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

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

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")

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADAE <- tmc_ex_adae
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADAE <- data[["ADAE"]]

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

```
    modules = modules(
      tm_t_events(
        label = "Adverse Event Table",
        dataname = "ADAE",
        arm_var = variables(
          choices = c("ARM", "ARMCD"),
          selected = "ARM"
        ),
        llt = variables(
          choices = c("AETERM", "AEDECOD"),
          selected = "AEDECOD"
        ),
        hlt = variables(
          choices = c("AEBODSYS", "AESOC"),
          selected = "AEBODSYS"
        ),
        add_total = TRUE,
        event_type = "adverse event"
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

`tm_t_events_by_grade` *teal Module: Events by Grade*

Description

This module produces a table to summarize events by grade.

Usage

```
tm_t_events_by_grade(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  hlt,
  llt,
  grade,
 grading_groups = list(`Any Grade (%)` = c("1", "2", "3", "4", "5"), `Grade 1-2 (%)` =
    c("1", "2"), `Grade 3-4 (%)` = c("3", "4"), `Grade 5 (%)` = "5"),
  col_by_grade = FALSE,
  prune_freq = 0,
  prune_diff = 0,
  add_total = TRUE,
```

```
  total_label = default_total_label(),
  na_level = tern::default_na_str(),
  drop_arm_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.
parentname	(character) parent analysis data used in teal module, usually this refers to ADSL.
arm_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) treatment arm variable on parentname.
hlt	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) high-level term variable on dataname (optional).
llt	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) low-level term variable on dataname (optional).
grade	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) grade variable on dataname.
grading_groups	(list) named list of grading groups used when col_by_grade = TRUE.
col_by_grade	(logical) whether to display the grading groups in nested columns.
prune_freq	(number) threshold to use for trimming table using event incidence rate in any column.
prune_diff	(number) threshold to use for trimming table using as criteria difference in rates between any two columns.
add_total	(logical) whether to include column with total number of patients.
total_label	(string) string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

na_level (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

drop_arm_levels (logical)
whether to drop unused levels of arm_var. If TRUE, arm_var levels are set to those used in the dataname dataset. If FALSE, arm_var levels are set to those used in the parentname dataset. If dataname and parentname are the same, then drop_arm_levels is set to TRUE and user input for this parameter is ignored.

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.

basic_table_args (basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- table (TableTree as created from rtables::build_table)

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_t_events_by_grade(
   ..., # arguments for module
```

```
    decorators = list(
      table = teal_transform_module(...) # applied only to `table` output
    )
)
```

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  library(teal.modules.clinical)
  library(teal.data)
  library(rtables)
  library(dplyr)
  ADSL <- tmc_ex_adsl
  .lbls_adae <- col_labels(tmc_ex_adae)
  ADAE <- tmc_ex_adae %>%
    mutate_if(is.character, as.factor) #' be certain of having factors
  col_labels(ADAE) <- .lbls_adae
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADAE <- data[["ADAE"]]

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

```
      tm_t_events_by_grade(
        label = "Adverse Events by Grade Table",
        dataname = "ADAE",
        arm_var = variables(choices = c("ARM", "ARMCD"), selected = "ARM"),
        llt = variables(
          choices = c("AETERM", "AEDECOD"),
          selected = "AEDECOD"
        ),
        hlt = variables(
          choices = c("AEBODSYS", "AESOC"),
          selected = "AEBODSYS"
        ),
        grade = variables(
          choices = c("AETOXGR", "AESEV"),
          selected = "AETOXGR"
        )
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

`tm_t_events_patyear` *teal Module: Event Rates Adjusted for Patient-Years*

Description

This module produces a table of event rates adjusted for patient-years.

Usage

```
tm_t_events_patyear(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  events_var,
  paramcd,
  aval_var = teal.picks::variables("AVAL", "AVAL", fixed = TRUE),
  avalu_var = teal.picks::variables("AVALU", "AVALU", fixed = TRUE),
  add_total = TRUE,
  total_label = default_total_label(),
  na_level = tern::default_na_str(),
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  drop_arm_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
```

```
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.
parentname	(character) parent analysis data used in teal module, usually this refers to ADSL.
arm_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) If there are two elements selected for arm_var, the second variable is nested under the first.
events_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for the variable with all event counts.
paramcd	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) variable used to filter the analysis dataset (typically PARAMCD). The values() element is added internally to allow users to pick parameter value(s) interactively.
aval_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and pre-selected option for the analysis variable.
avalu_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for the analysis unit variable.
add_total	(logical) whether to include column with total number of patients.
total_label	(string) string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").
na_level	(string) used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").
conf_level	(teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted) available confidence levels and default selection, each in the range (0, 1).

drop_arm_levels
: (logical)
whether to drop unused levels of arm_var. If TRUE, arm_var levels are set to those used in the dataname dataset. If FALSE, arm_var levels are set to those used in the parentname dataset. If dataname and parentname are the same, then drop_arm_levels is set to TRUE and user input for this parameter is ignored.

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.

basic_table_args
: (basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- table (ElementaryTable as created from rtables::build_table)

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

example-2 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(dplyr)

data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADAETTE <- tmc_ex_adaette %>%
    filter(PARAMCD %in% c("AETTE1", "AETTE2", "AETTE3")) %>%
    mutate(is_event = CNSR == 0) %>%
    mutate(n_events = as.integer(is_event))
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADAETTE <- data[["ADAETTE"]]

# 1. Basic Example

app <- init(
  data = data,
  modules = modules(
    tm_t_events_patyear(
      label = "AE Rate Adjusted for Patient-Years At Risk Table",
      dataname = "ADAETTE",
      arm_var = variables(choices = c("ARM", "ARMCD")),
      add_total = TRUE,
```

```
      events_var = variables(choices = "n_events"),
      paramcd = variables(choices = "PARAMCD"),
      aval_var = variables(choices = "AVAL", fixed = TRUE),
      avalu_var = variables(choices = "AVALU", fixed = TRUE)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}

# 2. Example with table split on 2 arm_var variables

app <- init(
  data = data,
  modules = modules(
    tm_t_events_patyear(
      label = "AE Rate Adjusted for Patient-Years At Risk Table",
      dataname = "ADAETTE",
      arm_var = variables(
        choices = c("ARM", "ARMCD", "SEX"),
        selected = c("ARM", "SEX")
      ),
      add_total = TRUE,
      events_var = variables(choices = "n_events"),
      paramcd = variables(choices = "PARAMCD"),
      aval_var = variables(choices = "AVAL", fixed = TRUE),
      avalu_var = variables(choices = "AVALU", fixed = TRUE)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_t_events_summary` *teal Module: Adverse Events Summary*

Description

This module produces an adverse events summary table.

Usage

```
tm_t_events_summary(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
```

```
  flag_var_anl = NULL,
  flag_var_aesi = NULL,
  dthfl_var = teal.picks::variables("DTHFL", "DTHFL", fixed = TRUE),
  dcsreas_var = teal.picks::variables("DCSREAS", "DCSREAS", fixed = TRUE),
  llt = teal.picks::variables("AEDECOD", "AEDECOD", fixed = TRUE),
  aeseq_var = teal.picks::variables("AESEQ", "AESEQ", fixed = TRUE),
  add_total = TRUE,
  total_label = default_total_label(),
  na_level = tern::default_na_str(),
  count_dth = TRUE,
  count_wd = TRUE,
  count_subj = TRUE,
  count_pt = TRUE,
  count_events = TRUE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
If there are two elements selected for `arm_var`, the second variable is nested under the first.

`flag_var_anl` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted or `NULL`)
vector with names of flag variables from `dataset` used to count adverse event sub-groups (e.g. Serious events, Related events, etc.). Variable labels are used as table row names if they exist.

`flag_var_aesi` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted or `NULL`)
vector with names of flag variables from `dataset` used to count adverse event special interest groups. All flag variables must be of type `logical`. Variable labels are used as table row names if they exist.

`dthfl_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as death flag variable. Records with ‘"Y"“ are summarized in the table row for "Total number of deaths".

dcsreas_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected with all available choices and preselected option for variable names that can be used as study discontinuation reason variable. Records with "ADVERSE EVENTS" are summarized in the table row for "Total number of patients withdrawn from study due to an AE".

llt
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
name of the variable with low level term for events.

aeseq_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
variable for adverse events sequence number from dataset. Used for counting total number of events.

add_total
: (logical)
whether to include column with total number of patients.

total_label
: (string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

na_level
: (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

count_dth
: (logical)
whether to show count of total deaths (based on dthfl_var). Defaults to TRUE.

count_wd
: (logical)
whether to show count of patients withdrawn from study due to an adverse event (based on dcsreas_var). Defaults to TRUE.

count_subj
: (logical)
whether to show count of unique subjects (based on USUBJID). Only applies if event flag variables are provided.

count_pt
: (logical)
whether to show count of unique preferred terms (based on llt). Only applies if event flag variables are provided.

count_events
: (logical)
whether to show count of events (based on aeseq_var). Only applies if event flag variables are provided.

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.

basic_table_args
: (basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the

module table. The argument is merged with option `teal.basic_table_args` and with default module arguments (hard coded in the module body). For more details, see the vignette: `vignette("custom-basic-table-arguments", package = "teal.widgets")`.

`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.

Decorating Module

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

- `table` (`TableTree` as created from `rtables::build_table`)

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  library(teal.modules.clinical)
  library(dplyr)
  library(tern)
  library(formatters)
  ADSL <- tmc_ex_adsl %>%
    mutate(
      DTHFL = case_when(
        !is.na(DTHDT) ~ "Y",
        TRUE ~ ""
      ) %>% with_label("Subject Death Flag")
    )
  ADAE <- tmc_ex_adae

  .add_event_flags <- function(dat) {
    dat <- dat %>%
      mutate(
        TMPFL_SER = AESER == "Y",
        TMPFL_REL = AEREL == "Y",
        TMPFL_GR5 = AETOXGR == "5",
        TMP_SMQ01 = !is.na(SMQ01NAM),
        TMP_SMQ02 = !is.na(SMQ02NAM),
        TMP_CQ01 = !is.na(CQ01NAM)
      )
    column_labels <- list(
      TMPFL_SER = "Serious AE",
      TMPFL_REL = "Related AE",
      TMPFL_GR5 = "Grade 5 AE",
      TMP_SMQ01 = aesi_label(dat[["SMQ01NAM"]], dat[["SMQ01SC"]]),
      TMP_SMQ02 = aesi_label("Y.9.9.9.9/Z.9.9.9.9 AESI"),
      TMP_CQ01 = aesi_label(dat[["CQ01NAM"]])
    )
    col_labels(dat)[names(column_labels)] <- as.character(column_labels)
    dat
  }

  #' Generating user-defined event flags.
  ADAE <- ADAE %>% .add_event_flags()

  .ae_anl_vars <- names(ADAE)[startsWith(names(ADAE), "TMPFL_")]
  .aesi_vars <- names(ADAE)[startsWith(names(ADAE), "TMP_")]
})
```

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

app <- init(
  data = data,
  modules = modules(
    tm_t_events_summary(
      label = "Adverse Events Summary",
      dataname = "ADAE",
      arm_var = variables(choices = c("ARM", "ARMCD")),
      flag_var_anl = variables(
        choices = data[[".ae_anl_vars"]],
        selected = data[[".ae_anl_vars"]][1],
        multiple = TRUE,
        ordered = TRUE,
        fixed = FALSE
      ),
      flag_var_aesi = variables(
        choices = data[[".aesi_vars"]],
        selected = data[[".aesi_vars"]][1],
        multiple = TRUE,
        ordered = TRUE,
        fixed = FALSE
      ),
      add_total = TRUE
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_t_exposure *teal Module: Exposure Table for Risk management plan*

Description

The module produces an exposure table for risk management plan.

Usage

```
tm_t_exposure(
  label,
  dataname,
  parentname = "ADSL",
  row_by_var,
  col_by_var,
  paramcd = teal.picks::variables("PARAMCD"),
  paramcd_label = "PARAM",
```

```
  id_var = teal.picks::variables("USUBJID", "USUBJID", fixed = TRUE),
  parcat,
  aval_var = teal.picks::variables("AVAL", "AVAL", fixed = TRUE),
  avalu_var = teal.picks::variables("AVALU", "AVALU", fixed = TRUE),
  add_total,
  total_label = default_total_label(),
  add_total_row = TRUE,
  total_row_label = "Total number of patients and patient time*",
  na_level = tern::default_na_str(),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label`	(`character`) menu item label of the module in the teal app.
`dataname`	(`character`) analysis data used in teal module.
`parentname`	(`character`) parent analysis data used in teal module, usually this refers to `ADSL`.
`row_by_var`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) variable names that can be used to split rows (`dataname`).
`col_by_var`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) variable names that can be used to split columns (`parentname`).
`paramcd`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) variable used to filter by parameter (`dataname`); `values()` is added internally.
`paramcd_label`	(`character`) the column from the dataset where the value will be used to label the argument `paramcd`.
`id_var`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) object specifying the variable name for subject id.
`parcat`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) parameter category column on `dataname`; `values()` is added internally.
`aval_var`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) object with all available choices and pre-selected option for the analysis variable.
`avalu_var`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)

object with all available choices and preselected option for the analysis unit variable.

add_total (logical)
whether to include column with total number of patients.

total_label (string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

add_total_row (flag)
whether a "total" level should be added after the others which includes all the levels that constitute the split. A custom label can be set for this level via the total_row_label argument.

total_row_label
(character)
string to display as total row label if row is enabled (see add_total_row).

na_level (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

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.

basic_table_args
(basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- `table` (`ElementaryTable` as created from `rtables::build_table`)

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(dplyr)

data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADEX <- tmc_ex_adex

  set.seed(1, kind = "Mersenne-Twister")
```

```
  .labels <- col_labels(ADEX, fill = FALSE)
  ADEX <- ADEX %>%
    distinct(USUBJID, .keep_all = TRUE) %>%
    mutate(
      PARAMCD = "TDURD",
      PARAM = "Overall duration (days)",
      AVAL = sample(x = seq(1, 200), size = n(), replace = TRUE),
      AVALU = "Days"
    ) %>%
    bind_rows(ADEX)
  col_labels(ADEX) <- .labels
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_t_exposure(
      label = "Duration of Exposure Table",
      dataname = "ADEX",
      paramcd = variables(choices = "PARAMCD"),
      col_by_var = variables(
        choices = c("SEX", "ARM"),
        selected = "SEX"
      ),
      row_by_var = variables(
        choices = c("RACE", "REGION1", "STRATA1", "SEX"),
        selected = "RACE"
      ),
      parcat = picks(variables(choices = "PARCAT2"), values(), check_dataset = FALSE),
      add_total = FALSE
    )
  ),
  filter = teal_slices(teal_slice("ADSL", "SAFFL", selected = "Y"))
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_t_glm_counts` *Teal Module: Regression Counts Summary*

Description

Summarize results of a Poisson negative binomial regression that is result of a generalized linear model of one (e.g. arm) or more covariates.

Usage

```
tm_t_glm_counts(
  label = "Counts Module",
  dataname,
  parentname = "ADSL",
  aval_var = teal.picks::variables("AVAL", "AVAL", fixed = TRUE),
  arm_var,
  strata_var,
  rate_mean_method = c("emmeans", "ppmeans"),
  distribution = c("negbin", "quasipoisson", "poisson"),
  offset_var,
  cov_var,
  arm_ref_comp = NULL,
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  add_total = FALSE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`aval_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `arm_var`. It defines the grouping variable in the results table.

`strata_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
names of the variables for stratified analysis.

`rate_mean_method`
(character) method used to estimate the mean odds ratio. Either "emmeans" or "ppmeans" (as in `summarize_glm_count()`).

`distribution` (character) value specifying the distribution used in the regression model (Poisson: `"poisson"`, Quasi-Poisson: `"quasipoisson"`, negative binomial: `"negbin"`).

offset_var (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
optional offset column (dataname).

cov_var (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the covariates variables.

arm_ref_comp (list) optional,
if specified it must be a named list with each element corresponding to an arm variable in ADSL and the element must be another list (possibly with delayed teal.picks::variables() or teal.picks::values(); legacy teal.transform::variable_choices and teal.transform::value_choices() are deprecated but still accepted) with the elements named ref and comp that define the default reference and comparison arms when the arm variable is changed.

conf_level (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
available confidence levels and default selection, each in the range (0, 1).

add_total (logical)
initial value for the "Add All Patients column" checkbox when comparing arms.

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.

basic_table_args
(basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Details

- Teal module for tern::summarize_glm_count() analysis, that summarizes results of a Poisson negative binomial regression.
- The arm and stratification variables are taken from the parentname data.

Value

a teal_module object.

Decorating Module

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

- table (TableTree - output of rtables::build_table())

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

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

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")

Examples in Shinylive

example-1 Open in Shinylive

See Also

summarize_glm_count()

Examples

```
data <- within(teal_data(), {
  ADSL <- tern::tern_ex_adsl
  ADTTE <- tern::tern_ex_adtte
})

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

arm_ref_comp <- list(
  ACTARMCD = list(
    ref = "ARM B",
    comp = c("ARM A", "ARM C")
  ),
  ARM = list(
    ref = "B: Placebo",
    comp = c("A: Drug X", "C: Combination")
  )
)

ADSL <- data[["ADSL"]]
ADTTE <- data[["ADTTE"]]
# Initialize the teal app
app <- init(
  data = data,
  modules = modules(
    tm_t_glm_counts(
      dataname = "ADTTE",
      arm_var = variables(
        choices = c("ARM", "ARMCD", "ACTARMCD"),
        selected = "ARMCD"
      ),
      arm_ref_comp = arm_ref_comp,
      aval_var = variables(choices = "AVAL", fixed = TRUE),
      strata_var = variables(choices = "SEX", selected = NULL),
      offset_var = variables(choices = "AGE", selected = NULL),
      cov_var = variables(choices = "SITEID", selected = NULL)
    )
  )
)

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

`tm_t_logistic` *teal Module: Logistic Regression*

Description

This module produces a multi-variable logistic regression table consistent with the TLG Catalog template `LGRT02` available here.

Usage

```
tm_t_logistic(
  label,
  dataname,
  parentname = "ADSL",
  arm_var = NULL,
  arm_ref_comp = NULL,
  paramcd,
  cov_var = NULL,
  avalc_var = teal.picks::variables("AVALC", "AVALC", fixed = TRUE),
  conf_level = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to ADSL.

`arm_var` (teal.picks::variables or `teal.transform::choices_selected()` or NULL)
object with all available choices and preselected option for variable names that can be used as `arm_var`. This defines the grouping variable(s) in the results table. If there are two elements selected for `arm_var`, the second variable will be nested under the first variable. If NULL, no arm/treatment variable is included in the logistic model.

`arm_ref_comp` (list) optional,
if specified it must be a named list with each element corresponding to an arm variable in ADSL and the element must be another list (possibly with delayed `teal.picks::variables()` or `teal.picks::values()`; legacy `teal.transform::variable_choices` and `teal.transform::value_choices()` are deprecated but still accepted) with the elements named `ref` and `comp` that define the default reference and comparison arms when the arm variable is changed.

`paramcd` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the parameter code variable from `dataname`.

`cov_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for the covariates variables.

avalc_var (teal.picks::variables or teal.transform::choices_selected())
object with all available choices and preselected option for the analysis variable (categorical).

conf_level (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
available confidence levels and default selection, each in the range (0, 1).

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.

basic_table_args
(basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- table (TableTree - output of rtables::build_table())

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
library(dplyr)

data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADRS <- tmc_ex_adrs %>%
    filter(PARAMCD %in% c("BESRSPI", "INVET"))
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADRS <- data[["ADRS"]]

arm_ref_comp <- list(
  ACTARMCD = list(
    ref = "ARM B",
    comp = c("ARM A", "ARM C")
  ),
  ARM = list(
    ref = "B: Placebo",
    comp = c("A: Drug X", "C: Combination")
  )
)

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

```
  modules = modules(
    tm_t_logistic(
      label = "Logistic Regression",
      dataname = "ADRS",
      paramcd = picks(variables("PARAMCD", "PARAMCD"),
        values(multiple = FALSE),
        check_dataset = FALSE
      ),
      cov_var = variables(selected = "ARM"),
      avalc_var = variables("AVALC", fixed = TRUE),
      conf_level = values(c(0.95, 0.9, 0.8), 0.95)
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_t_mult_events` *teal Module: Multiple Events by Term*

Description

This module produces a table of multiple events by term.

Usage

```
tm_t_mult_events(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  seq_var,
  hlt,
  llt,
  add_total = TRUE,
  total_label = default_total_label(),
  na_level = tern::default_na_str(),
  event_type = "event",
  title_text = "Concomitant Medications",
  drop_arm_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to ADSL.

`arm_var` (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable in the results table.

`seq_var` (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as analysis sequence number variable. Used for counting the unique number of events.

`hlt` (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
name of the variable with high level term for events.

`llt` (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
name of the variable with low level term for events.

`add_total` (logical)
whether to include column with total number of patients.

`total_label` (string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

`na_level` (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

`event_type` (character)
type of event that is summarized (e.g. adverse event, treatment). Default is "event".

`title_text` (string)
text to display as the first part of the dynamic table title. The table title is constructed as follows: "title_text by hlt and llt". Defaults to "Concomitant Medications".

`drop_arm_levels` (logical)
whether to drop unused levels of arm_var. If TRUE, arm_var levels are set to those used in the dataname dataset. If FALSE, arm_var levels are set to those used in the parentname dataset. If dataname and parentname are the same, then drop_arm_levels is set to TRUE and user input for this parameter is ignored.

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.

basic_table_args (basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- table (TableTree - output of rtables::build_table())

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADCM <- tmc_ex_adcm
})
join_keys(data) <- default_cdisc_join_keys[names(data)]
adcm_keys <- c("STUDYID", "USUBJID", "ASTDTM", "CMSEQ", "ATC1", "ATC2", "ATC3", "ATC4")
join_keys(data)["ADCM", "ADCM"] <- adcm_keys

ADSL <- data[["ADSL"]]
ADCM <- data[["ADCM"]]

app <- init(
  data = data,
  modules = modules(
    tm_t_mult_events(
      label = "Concomitant Medications by Medication Class and Preferred Name",
      dataname = "ADCM",
      arm_var = variables(choices = c("ARM", "ARMCD"), selected = "ARM"),
      seq_var = variables(choices = "CMSEQ", selected = "CMSEQ", fixed = TRUE),
      hlt = variables(
        choices = c("ATC1", "ATC2", "ATC3", "ATC4"),
        selected = c("ATC1", "ATC2", "ATC3", "ATC4")
      ),
      llt = variables(
        choices = "CMDECOD",
        selected = "CMDECOD"
      ),
      add_total = TRUE,
      event_type = "treatment"
    )
  )
```

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

---

tm_t_pp_basic_info *teal Module: Patient Profile Basic Info*

Description

This module produces a patient profile basic info report using ADaM datasets.

Usage

```
tm_t_pp_basic_info(
  label,
  dataname = "ADSL",
  patient_col = "USUBJID",
  vars = NULL,
  pre_output = NULL,
  post_output = NULL,
  transformators = list(),
  decorators = lifecycle::deprecated()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.
patient_col	(character) name of patient ID variable.
vars	(teal.picks::variables or teal.transform::choices_selected) object specifying available choices and preselected variables from dataname to show in the table.
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.

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")`

Examples in Shinylive

example-1 Open in Shinylive

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_t_pp_basic_info(
      label = "Basic Info",
      dataname = "ADSL",
      patient_col = "USUBJID",
      vars = variables(
        c("ARM", "AGE", "SEX", "COUNTRY", "RACE", "EOSSTT"),
        multiple = TRUE
      )
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

tm_t_pp_laboratory *teal Module: Patient Profile Laboratory Table*

Description

This module produces a patient profile laboratory table using ADaM datasets.

Usage

```
tm_t_pp_laboratory(
  label,
  dataname = "ADLB",
  parentname = "ADSL",
  patient_col = "USUBJID",
  time_points = NULL,
  aval = lifecycle::deprecated(),
  aval_var = NULL,
  avalu = lifecycle::deprecated(),
  avalu_var = NULL,
  param = NULL,
  paramcd = NULL,
  anrind = NULL,
  pre_output = NULL,
  post_output = NULL,
  transformators = list(),
  decorators = lifecycle::deprecated()
)
```

Arguments

`label`	(`character`) menu item label of the module in the teal app.
`dataname`	(`character`) analysis data used in teal module.
`parentname`	(`character`) parent analysis data used in teal module, usually this refers to `ADSL`.
`patient_col`	(`character`) name of patient ID variable.
`time_points`	(teal.picks::variables or `teal.transform::choices_selected()`) object with all available choices and preselected option for the time variable from `dataname`.
`aval`	**[Deprecated]** Please use the `aval_var` argument instead.
`aval_var`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) object with all available choices and pre-selected option for the analysis variable.

avalu	**[Deprecated]** Please use the avalu_var argument instead.
avalu_var	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for the analysis unit variable.
param	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the PARAM variable from dataname.
paramcd	(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted) object with all available choices and preselected option for the parameter code variable from dataname.
anrind	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the ANRIND variable from dataname. Variable should have the following 3 levels: "HIGH", "LOW", and "NORMAL".
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.

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")

Examples in Shinylive

example-1 Open in Shinylive

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADLB <- tmc_ex_adlb
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADLB <- data[["ADLB"]]

app <- init(
  data = data,
  modules = modules(
    tm_t_pp_laboratory(
      label = "Vitals",
      dataname = "ADLB",
      patient_col = "USUBJID",
      time_points = variables("ADY", fixed = TRUE),
      aval_var = variables("AVAL", fixed = TRUE),
      avalu_var = variables("AVALU", fixed = TRUE),
      param = variables("PARAM", fixed = TRUE),
      paramcd = variables("PARAMCD", fixed = TRUE),
      anrind = variables("ANRIND", fixed = TRUE),
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_t_pp_medical_history

teal Module: Patient Profile Medical History

Description

This module produces a patient profile medical history report using ADaM datasets.

Usage

```
tm_t_pp_medical_history(
  label,
  dataname = "ADMH",
  parentname = "ADSL",
  patient_col = "USUBJID",
  mhterm = NULL,
  mhbodsys = NULL,
```

```
  mhdistat = NULL,
  pre_output = NULL,
  post_output = NULL,
  transformators = list(),
  decorators = list()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.
parentname	(character) parent analysis data used in teal module, usually this refers to ADSL.
patient_col	(character) name of patient ID variable.
mhterm	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the MHTERM variable from dataname.
mhbodsys	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the MHBODSYS variable from dataname.
mhdistat	(teal.picks::variables or teal.transform::choices_selected()) object with all available choices and preselected option for the MHDISTAT variable from dataname.
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.

Decorating Module

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

- `table` (`TableTree` - output of `rtables::build_table()`)

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADMH <- tmc_ex_admh
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]
ADMH <- data[["ADMH"]]

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

```
      tm_t_pp_medical_history(
        label = "Medical History",
        dataname = "ADMH",
        parentname = "ADSL",
        patient_col = "USUBJID",
        mhterm = variables("MHTERM", fixed = TRUE),
        mhbodsys = variables("MHBODSYS", fixed = TRUE),
        mhdistat = variables("MHDISTAT", fixed = TRUE),
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

---

tm_t_pp_prior_medication

teal Module: Patient Profile Prior Medication

Description

This module produces a patient profile prior medication report using ADaM datasets.

Usage

```
tm_t_pp_prior_medication(
  label,
  dataname = "ADCM",
  parentname = "ADSL",
  patient_col = "USUBJID",
  atirel = NULL,
  cmdecod = NULL,
  cmindc = NULL,
  cmstdy = NULL,
  pre_output = NULL,
  post_output = NULL,
  transformators = list(),
  decorators = lifecycle::deprecated()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.

parentname (character)
parent analysis data used in teal module, usually this refers to ADSL.

patient_col (character)
name of patient ID variable.

atirel (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the ATIREL variable from dataname.

cmdecod (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMDECOD variable from dataname.

cmindc (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMINDC variable from dataname.

cmstdy (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the CMSTDY variable from dataname.

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.

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")

Examples in Shinylive

example-1 Open in Shinylive

Examples

```
library(dplyr)
data <- teal_data()
data <- within(data, {
  ADCM <- tmc_ex_adcm
  ADSL <- tmc_ex_adsl %>% filter(USUBJID %in% ADCM$USUBJID)
  ADCM$CMASTDTM <- ADCM$ASTDTM
  ADCM$CMAENDTM <- ADCM$AENDTM
})
join_keys(data) <- default_cdisc_join_keys[names(data)]
adcm_keys <- c("STUDYID", "USUBJID", "ASTDTM", "CMSEQ", "ATC1", "ATC2", "ATC3", "ATC4")
join_keys(data)["ADCM", "ADCM"] <- adcm_keys

ADSL <- data[["ADSL"]]
ADCM <- data[["ADCM"]]

app <- init(
  data = data,
  modules = modules(
    tm_t_pp_prior_medication(
      label = "Prior Medication",
      dataname = "ADCM",
      parentname = "ADSL",
      patient_col = "USUBJID",
      atirel = variables("ATIREL", fixed = TRUE),
      cmdecod = variables("CMDECOD", fixed = TRUE),
      cmindc = variables("CMINDC", fixed = TRUE),
      cmstdy = variables("ASTDY", fixed = TRUE),
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_t_shift_by_arm *teal Module: Shift by Arm*

Description

This module produces a summary table of analysis indicator levels by arm.

Usage

```
tm_t_shift_by_arm(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  paramcd,
  visit_var,
  aval_var,
  base_var = lifecycle::deprecated(),
  baseline_var,
  treatment_flag_var = teal.picks::variables("ONTRTFL", "ONTRTFL"),
  treatment_flag = teal.picks::values("Y", "Y", fixed = TRUE, multiple = FALSE),
  useNA = c("ifany", "no"),
  na_level = tern::default_na_str(),
  add_total = FALSE,
  total_label = default_total_label(),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `arm_var`. It defines the grouping variable in the results table.

`paramcd` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
variable for lab parameter code. The `values()` element is added internally to allow users to filter the parameter values interactively.

`visit_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
variable for analysis visit. The `values()` element is added internally to allow users to filter the visit values interactively.

`aval_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

base_var **[Deprecated]** Please use the baseline_var argument instead.

baseline_var (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for variable values that can be used as baseline_var.

treatment_flag_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
on treatment flag variable.

treatment_flag (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted) value matching treatment_flag_var for on-treatment records (default "Y").

useNA (character)
whether missing data (NA) should be displayed as a level.

na_level (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

add_total (logical)
whether to include column with total number of patients.

total_label (string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

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.

basic_table_args
(basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- table (TableTree - output of rtables::build_table())

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

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

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")

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADEG <- tmc_ex_adeg
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_t_shift_by_arm(
      label = "Shift by Arm Table",
      dataname = "ADEG",
      arm_var = variables(choices = c("ARM", "ARMCD")),
      paramcd = variables(choices = "PARAMCD"),
      visit_var = variables(choices = "AVISIT"),
      aval_var = variables(choices = "ANRIND"),
      baseline_var = variables(choices = "BNRIND"),
      useNA = "ifany"
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_t_shift_by_arm_by_worst

teal Module: Shift by Arm by Worst Analysis Indicator Level

Description

This module produces a summary table of worst analysis indicator variable level per subject by arm.

Usage

```
tm_t_shift_by_arm_by_worst(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  paramcd,
  aval_var,
  base_var = lifecycle::deprecated(),
  baseline_var,
  worst_flag_var,
```

```
  worst_flag,
  treatment_flag_var = teal.picks::variables("ONTRTFL", "ONTRTFL"),
  treatment_flag = teal.picks::values("Y", "Y", fixed = TRUE, multiple = FALSE),
  useNA = c("ifany", "no"),
  na_level = tern::default_na_str(),
  add_total = FALSE,
  total_label = default_total_label(),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `arm_var`. It defines the grouping variable in the results table.

`paramcd` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
variable for lab parameter code. The `values()` element is added internally to allow users to filter the parameter values interactively.

`aval_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

`base_var` **[Deprecated]** Please use the `baseline_var` argument instead.

`baseline_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable values that can be used as `baseline_var`.

`worst_flag_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as worst flag variable.

`worst_flag` (character)
value indicating worst analysis indicator level.

treatment_flag_var
: (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
on treatment flag variable.

treatment_flag
: (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
value matching treatment_flag_var for on-treatment records (default "Y").

useNA
: (character)
whether missing data (NA) should be displayed as a level.

na_level
: (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

add_total
: (logical)
whether to include column with total number of patients.

total_label
: (string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

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.

basic_table_args
: (basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- `table` (`TableTree` - output of `rtables::build_table()`)

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADEG <- tmc_ex_adeg
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_t_shift_by_arm_by_worst(
      label = "Shift by Arm Table",
      dataname = "ADEG",
```

```
        arm_var = variables(choices = c("ARM", "ARMCD")),
        paramcd = variables(choices = "PARAMCD"),
        worst_flag_var = variables(
          choices = c("WORS01FL", "WORS02FL"),
          selected = "WORS02FL"
        ),
        worst_flag = values("Y", "Y", fixed = TRUE),
        aval_var = variables(choices = c("AVALC", "ANRIND"), selected = "ANRIND"),
        baseline_var = variables(choices = c("BASEC", "BNRIND"), selected = "BNRIND"),
        useNA = "ifany"
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

`tm_t_shift_by_grade` *teal Module: Grade Summary Table*

Description

This module produces a summary table of worst grades per subject by visit and parameter.

Usage

```
tm_t_shift_by_grade(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  visit_var = teal.picks::variables("AVISIT", "AVISIT", fixed = TRUE),
  paramcd,
 worst_flag_var = teal.picks::variables(c("WGRLOVFL", "WGRLOFL", "WGRHIVFL", "WGRHIFL"),
    "WGRLOVFL"),
 worst_flag_indicator = teal.picks::values("Y", "Y", fixed = TRUE, multiple = FALSE),
  anl_toxgrade_var = teal.picks::variables("ATOXGR", "ATOXGR", fixed = TRUE),
  base_toxgrade_var = teal.picks::variables("BTOXGR", "BTOXGR", fixed = TRUE),
  id_var = teal.picks::variables("USUBJID", "USUBJID", fixed = TRUE),
  add_total = FALSE,
  total_label = default_total_label(),
  drop_arm_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
  na_level = tern::default_na_str(),
  code_missing_baseline = FALSE,
  basic_table_args = teal.widgets::basic_table_args(),
```

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

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `arm_var`. It defines the grouping variable in the results table.

`visit_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `visit` variable. Must be a factor in `dataname`.

`paramcd` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
variable for lab parameter code. The `values()` element is added internally to allow users to filter the parameter values interactively.

`worst_flag_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
variable for worst grade flag.

`worst_flag_indicator`
(`teal.picks::values()`; legacy `teal.transform::choices_selected()` is deprecated but still accepted)
value(s) indicating worst grade records.

`anl_toxgrade_var`
(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) variable for analysis toxicity grade.

`base_toxgrade_var`
(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) variable for baseline toxicity grade.

`id_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object specifying the variable name for subject id.

`add_total` (logical)
whether to include column with total number of patients.

`total_label` (string)
string to display as total column/row label if column/row is enabled (see `add_total`). Defaults to `"All Patients"`. To set a new default `total_label` to apply in all modules, run `set_default_total_label("new_default")`.

drop_arm_levels
: (logical)
whether to drop unused levels of arm_var. If TRUE, arm_var levels are set to those used in the dataname dataset. If FALSE, arm_var levels are set to those used in the parentname dataset. If dataname and parentname are the same, then drop_arm_levels is set to TRUE and user input for this parameter is ignored.

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.

na_level
: (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

code_missing_baseline
: (logical)
whether missing baseline grades should be counted as grade 0.

basic_table_args
: (basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- table (TableTree - output of rtables::build_table())

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADLB <- tmc_ex_adlb
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_t_shift_by_grade(
      label = "Grade Laboratory Abnormality Table",
      dataname = "ADLB",
      arm_var = variables(c("ARM", "ARMCD")),
    paramcd = picks(variables("PARAMCD"), values(selected = "ALT"), check_dataset = FALSE),
      worst_flag_var = variables(
        choices = c("WGRLOVFL", "WGRLOFL", "WGRHIVFL", "WGRHIFL"),
        selected = "WGRLOVFL"
```

```
        ),
        add_total = FALSE
      )
    ),
    filter = teal_slices(teal_slice("ADSL", "SAFFL", selected = "Y"))
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

---

tm_t_smq *teal Module: Adverse Events Table by Standardized MedDRA Query*

Description

This module produces an adverse events table by Standardized MedDRA Query.

Usage

```
tm_t_smq(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  id_var = teal.picks::variables("USUBJID", "USUBJID", fixed = TRUE),
  llt,
  add_total = TRUE,
  total_label = default_total_label(),
  sort_criteria = c("freq_desc", "alpha"),
  drop_arm_levels = TRUE,
  na_level = tern::default_na_str(),
  smq_varlabel = "Standardized MedDRA Query",
  baskets,
  scopes,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

label	(character) menu item label of the module in the teal app.
dataname	(character) analysis data used in teal module.

parentname (character)
parent analysis data used in teal module, usually this refers to ADSL.

arm_var (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
If there are two elements selected for arm_var, the second variable is nested under the first.

id_var (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object specifying the variable name for subject id.

llt (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
name of the variable with low level term for events.

add_total (logical)
whether to include column with total number of patients.

total_label (string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

sort_criteria (character)
how to sort the final table. Default option freq_desc sorts on column sort_freq_col by decreasing number of patients with event. Alternative option alpha sorts events alphabetically.

drop_arm_levels
(logical)
whether to drop unused levels of arm_var. If TRUE, arm_var levels are set to those used in the dataname dataset. If FALSE, arm_var levels are set to those used in the parentname dataset. If dataname and parentname are the same, then drop_arm_levels is set to TRUE and user input for this parameter is ignored.

na_level (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

smq_varlabel (character)
label to use for new column SMQ created by tern::h_stack_by_baskets().

baskets (teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected options for standardized/customized queries.

scopes (teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
object with all available choices for the scopes of standardized queries.

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.

basic_table_args
(basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- table (TableTree - output of rtables::build_table())

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

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

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")

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  library(teal.modules.clinical)
  library(dplyr)
  library(rtables)
  ADSL <- tmc_ex_adsl
  ADAE <- tmc_ex_adae

  .names_baskets <- grep("^(SMQ|CQ).*NAM$", names(ADAE), value = TRUE)
  .names_scopes <- grep("^SMQ.*SC$", names(ADAE), value = TRUE)
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_t_smq(
      label = "Adverse Events by SMQ Table",
      dataname = "ADAE",
      arm_var = variables(choices = c("ARM", "ARMCD", "SEX")),
      add_total = FALSE,
      baskets = variables(choices = starts_with("SMQ") | starts_with("CQ")),
      scopes = variables(choices = ends_with("SC")),
      llt = variables(choices = "AEDECOD")
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_t_summary *teal Module: Summary of Variables*

Description

This module produces a table to summarize variables.

Usage

```
tm_t_summary(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  summarize_vars,
  add_total = TRUE,
  total_label = default_total_label(),
  show_arm_var_labels = TRUE,
  useNA = c("ifany", "no"),
  na_level = tern::default_na_str(),
 numeric_stats = c("n", "mean_sd", "mean_ci", "median", "median_ci", "quantiles",
    "range", "geom_mean"),
  numeric_formats = NULL,
  denominator = c("N", "n", "omit"),
  drop_arm_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label`	(`character`) menu item label of the module in the teal app.
`dataname`	(`character`) analysis data used in teal module.
`parentname`	(`character`) parent analysis data used in teal module, usually this refers to `ADSL`.
`arm_var`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) If there are two elements selected for `arm_var`, the second variable is nested under the first.
`summarize_vars`	(`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted) names of the variables that should be summarized.
`add_total`	(`logical`) whether to include column with total number of patients.
`total_label`	(`string`) string to display as total column/row label if column/row is enabled (see `add_total`).

Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

show_arm_var_labels
: (flag)
whether arm variable label(s) should be displayed. Defaults to TRUE.

useNA
: (character)
whether missing data (NA) should be displayed as a level.

na_level
: (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

numeric_stats
: (character)
names of statistics to display for numeric summary variables. Available statistics are n, mean_sd, mean_ci, median, median_ci, quantiles, range, and geom_mean.

numeric_formats
: (named list or NULL)
format patterns for numeric statistics. Names should match the statistics in numeric_stats. If NULL, defaults from tern::analyze_vars() are used.

denominator
: (character)
chooses how percentages are calculated. With option N, the reference population from the column total is used as the denominator. With option n, the number of non-missing records in this row and column intersection is used as the denominator. If omit is chosen, then the percentage is omitted.

drop_arm_levels
: (logical)
whether to drop unused levels of arm_var. If TRUE, arm_var levels are set to those used in the dataname dataset. If FALSE, arm_var levels are set to those used in the parentname dataset. If dataname and parentname are the same, then drop_arm_levels is set to TRUE and user input for this parameter is ignored.

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.

basic_table_args
: (basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- `table` (`TableTree` - output of `rtables::build_table()`)

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
# Preparation of the test case - use `EOSDY` and `DCSREAS` variables to demonstrate missing data.
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADSL$EOSDY[1] <- NA_integer_
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

ADSL <- data[["ADSL"]]

app <- init(
  data = data,
  modules = modules(
    tm_t_summary(
      label = "Demographic Table",
      dataname = "ADSL",
      arm_var = variables(choices = c("ARM", "ARMCD"), selected = "ARM"),
      add_total = TRUE,
      summarize_vars = variables(
        choices = c("SEX", "RACE", "BMRKR2", "EOSDY", "DCSREAS", "AGE"),
        selected = c("SEX", "RACE")
      ),
      useNA = "ifany"
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

`tm_t_summary_by` *teal Module: Summarize Variables by Row Groups*

Description

This module produces a table to summarize variables by row groups.

Usage

```
tm_t_summary_by(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  by_vars,
  summarize_vars,
  id_var = teal.picks::variables("USUBJID", "USUBJID", fixed = TRUE),
```

```
  paramcd = NULL,
  add_total = TRUE,
  total_label = default_total_label(),
  parallel_vars = FALSE,
  row_groups = FALSE,
  useNA = c("ifany", "no"),
  na_level = tern::default_na_str(),
  numeric_stats = c("n", "mean_sd", "median", "range"),
  categorical_stats = c("n", "count"),
  denominator = teal.picks::values(c("n", "N", "omit"), "omit", fixed = TRUE),
  drop_arm_levels = TRUE,
  drop_zero_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
If there are two elements selected for `arm_var`, the second variable is nested under the first.

`by_vars` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names used to split the summary by rows.

`summarize_vars` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
names of the variables that should be summarized.

`id_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object specifying the variable name for subject id.

`paramcd` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
optional variable for parameter code filter. When provided, a `values()` selector is added with `multiple = FALSE`, so one parameter level is selected by default.

`add_total` (logical)
whether to include column with total number of patients.

total_label (string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

parallel_vars (logical)
whether summarized variables should be arranged in columns. Can only be set to TRUE if all chosen analysis variables are numeric.

row_groups (logical)
whether summarized variables should be arranged in row groups.

useNA (character)
whether missing data (NA) should be displayed as a level.

na_level (string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

numeric_stats (character)
names of statistics to display for numeric summary variables. Available statistics are n, mean_sd, mean_ci, median, median_ci, quantiles, range, and geom_mean.

categorical_stats
(character)
names of statistics to display for non-numeric summary variables. Available statistics are n, count, count_fraction, count_fraction_fixed_dp, fraction and n_blq.

denominator (character)
chooses how percentages are calculated. With option N, the reference population from the column total is used as the denominator. With option n, the number of non-missing records in this row and column intersection is used as the denominator. If omit is chosen, then the percentage is omitted.

drop_arm_levels
(logical)
whether to drop unused levels of arm_var. If TRUE, arm_var levels are set to those used in the dataname dataset. If FALSE, arm_var levels are set to those used in the parentname dataset. If dataname and parentname are the same, then drop_arm_levels is set to TRUE and user input for this parameter is ignored.

drop_zero_levels
(logical)
whether rows with zero counts in all columns should be removed from the table.

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.

basic_table_args
: (basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Decorating Module

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

- table (TableTree - output of rtables::build_table())

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

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

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")

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADLB <- tmc_ex_adlb
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

app <- init(
  data = data,
  modules = modules(
    tm_t_summary_by(
      label = "Summary by Row Groups Table",
      dataname = "ADLB",
      arm_var = variables(choices = c("ARM", "ARMCD"), multiple = TRUE),
      add_total = TRUE,
    by_vars = variables(choices = c("PARAM", "AVISIT"), selected = "AVISIT", multiple = TRUE),
    summarize_vars = variables(choices = c("AVAL", "CHG"), selected = "AVAL", multiple = TRUE),
      useNA = "ifany",
      paramcd = picks(
        variables(choices = "PARAMCD"),
        values(selected = "ALT", multiple = TRUE),
        check_dataset = FALSE
      )
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_t_tte *teal Module: Time-To-Event Table*

Description

This module produces a time-to-event analysis summary table, consistent with the TLG Catalog template for `TTET01` available here.

Usage

```
tm_t_tte(
  label,
  dataname,
  parentname = "ADSL",
  arm_var,
  arm_ref_comp = NULL,
  paramcd,
  strata_var,
  aval_var = teal.picks::variables("AVAL", "AVAL", fixed = TRUE),
  cnsr_var = teal.picks::variables("CNSR", "CNSR", fixed = TRUE),
  conf_level_coxph = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  conf_level_survfit = teal.picks::values(c(0.95, 0.9, 0.8), 0.95),
  time_points,
  time_unit_var = teal.picks::variables("AVALU", "AVALU", fixed = TRUE),
  event_desc_var = teal.picks::variables("EVNTDESC", "EVNTDESC", fixed = TRUE),
  add_total = FALSE,
  total_label = default_total_label(),
  na_level = tern::default_na_str(),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args(),
  transformators = list(),
  decorators = list()
)
```

Arguments

`label` (character)
menu item label of the module in the teal app.

`dataname` (character)
analysis data used in teal module.

`parentname` (character)
parent analysis data used in teal module, usually this refers to `ADSL`.

`arm_var` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)
object with all available choices and preselected option for variable names that can be used as `arm_var`. It defines the grouping variable in the results table.

`arm_ref_comp` (`list`) optional,
if specified it must be a named list with each element corresponding to an arm variable in `ADSL` and the element must be another list (possibly with delayed `teal.picks::variables()` or `teal.picks::values()`; legacy `teal.transform::variable_choices` and `teal.transform::value_choices()` are deprecated but still accepted) with the elements named `ref` and `comp` that define the default reference and comparison arms when the arm variable is changed.

`paramcd` (`teal.picks::variables()`; legacy `teal.transform` objects are deprecated but still accepted)

object with all available choices and preselected option for the parameter code variable from dataname.

strata_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
names of the variables for stratified analysis.

aval_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and pre-selected option for the analysis variable.

cnsr_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and preselected option for the censoring variable.

conf_level_coxph
(teal.transform::choices_selected())
object with all available choices and pre-selected option for confidence level, each within range of (0, 1).

conf_level_survfit
(teal.transform::choices_selected())
object with all available choices and pre-selected option for confidence level, each within range of (0, 1).

time_points
(teal.picks::values(); legacy teal.transform::choices_selected() is deprecated but still accepted)
object with all available choices and preselected option for time points that can be used in tern::surv_timepoint().

time_unit_var
(teal.picks::variables(); legacy teal.transform objects are deprecated but still accepted)
object with all available choices and pre-selected option for the time unit variable.

event_desc_var
(teal.picks::variables(), legacy teal.transform::choices_selected() objects are deprecated but still accepted
variable for event description.

add_total
(logical)
whether to include column with total number of patients.

total_label
(string)
string to display as total column/row label if column/row is enabled (see add_total). Defaults to "All Patients". To set a new default total_label to apply in all modules, run set_default_total_label("new_default").

na_level
(string)
used to replace all NA or empty values in character or factor variables in the data. Defaults to "<Missing>". To set a default na_level to apply in all modules, run set_default_na_str("new_default").

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.

basic_table_args
(basic_table_args) optional
object created by teal.widgets::basic_table_args() with settings for the module table. The argument is merged with option teal.basic_table_args and with default module arguments (hard coded in the module body). For more details, see the vignette: vignette("custom-basic-table-arguments", package = "teal.widgets").

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.

Details

- The core functionality of this module is based on tern::coxph_pairwise(), tern::surv_timepoint(), and tern::surv_time() from the tern package.
- The arm and stratification variables are taken from the parentname data.
- The following variables are used in the module:
  - AVAL: time to event
  - CNSR: 1 if record in AVAL is censored, 0 otherwise
  - PARAMCD: variable used to filter for endpoint (e.g. OS). After filtering for PARAMCD one observation per patient is expected

Value

a teal_module object.

Decorating Module

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

- table (TableTree - output of rtables::build_table())

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

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

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")`

Examples in Shinylive

example-1 Open in Shinylive

See Also

The TLG Catalog where additional example apps implementing this module can be found.

Examples

```
data <- teal_data()
data <- within(data, {
  ADSL <- tmc_ex_adsl
  ADTTE <- tmc_ex_adtte
})
join_keys(data) <- default_cdisc_join_keys[names(data)]

arm_ref_comp <- list(
  ACTARMCD = list(
    ref = "ARM B",
    comp = c("ARM A", "ARM C")
  ),
  ARM = list(
    ref = "B: Placebo",
    comp = c("A: Drug X", "C: Combination")
  )
)

app <- init(
  data = data,
  modules = modules(
    tm_t_tte(
      label = "Time To Event Table",
      dataname = "ADTTE",
      arm_var = variables(c("ARM", "ARMCD", "ACTARMCD")),
      arm_ref_comp = arm_ref_comp,
```

```
        paramcd = variables(c("PARAMCD", "PARAM")),
        strata_var = variables(c("SEX", "BMRKR2"), "SEX"),
        time_points = teal.picks::values(c(182, 243), 182)
      )
    )
  )
  if (interactive()) {
    shinyApp(app$ui, app$server)
  }
```

Index

∗ **datasets**
ex_data, 15

add_expr, 5
as_num, 6

base::factor(), *96*
base::Paren(), *7*, *21*
bracket_expr, 7

call_concatenate, 8
clean_description, 9
color_lab_values, 9
column_annotation_label, 10
control_coxph_annot(), *59*
control_surv_med_annot(), *59*
cs_to_des_filter, 11
cs_to_des_select, 12
cs_to_filter_spec, 12
cs_to_select_spec, 13

default_total_label, 14

ex_data, 15
extract_input, 16

get_var_labels, 17
ggplot2::ggplot(), *33*, *38*, *52*, *62*, *66*, *70*, *74*, *78*

h_concat_expr, 17

is.cs_or_des, 18

pipe_expr, 19
prepare_arm, 19
prepare_arm_levels, 21

set_default_total_label (default_total_label), 14
set_default_total_label(), *14*
shiny::helpText(), *26*, *30*, *34*, *40*, *45*, *50*, *54*, *59*, *64*, *68*, *72*, *76*, *79*, *83*, *87*, *91*, *96*, *101*, *107*, *111*, *115*, *119*, *124*, *128*, *132*, *136*, *138*, *141*, *143*, *146*, *149*, *153*, *157*, *161*, *164*, *168*, *173*
shiny::selectInput(), *11–13*
split_choices, 22
split_col_expr, 23
split_interactions, 24

teal.modules.clinical (teal.modules.clinical-package), 4
teal.modules.clinical-package, 4
teal.picks::picks(), *35*, *39*, *40*, *53*, *63*
teal.picks::values, *29*
teal.picks::values(), *25*, *26*, *29*, *40*, *44*, *49*, *50*, *53*, *58*, *63*, *83*, *87*, *90*, *91*, *95*, *100*, *101*, *114*, *128*, *131*, *132*, *149*, *153*, *156*, *160*, *171*, *172*
teal.picks::variables, *67*, *79*, *131*, *132*, *140*, *141*, *143*
teal.picks::variables(), *25*, *29*, *39–41*, *43*, *44*, *49*, *50*, *53*, *54*, *58*, *63*, *71*, *75*, *76*, *79*, *82*, *83*, *86*, *87*, *90*, *91*, *95*, *100*, *106*, *110*, *114*, *118*, *119*, *123*, *127*, *128*, *131*, *135*, *140*, *141*, *146*, *148*, *149*, *152*, *153*, *156*, *160*, *163*, *167*, *171*, *172*
teal.transform::choices_selected(), *11–13*, *18*, *22*, *29*, *67*, *79*, *131*, *132*, *140*, *141*, *143*, *172*
teal.transform::data_extract_spec(), *11*, *12*
teal.transform::filter_spec(), *13*
teal.transform::value_choices(), *25*, *29*, *44*, *49*, *53*, *58*, *90*, *95*, *100*, *128*, *131*, *171*
teal.transform::variable_choices(), *25*, *29*, *44*, *49*, *58*, *90*, *95*, *100*, *128*, *131*,

171
teal.widgets::basic_table_args(), 26, 30, 83, 87, 91, 96, 101, 107, 111, 115, 119, 124, 128, 132, 136, 149, 153, 157, 161, 164, 169, 173
teal.widgets::ggplot2_args(), 30, 34, 40, 45, 50, 54, 64, 68, 72, 76, 79
teal::teal_transform_module(), 26, 31, 35, 41, 46, 51, 55, 60, 65, 68, 73, 77, 80, 84, 88, 92, 97, 102, 108, 112, 116, 120, 125, 129, 133, 136, 144, 150, 154, 158, 161, 165, 169, 174
tern::analyze_vars(), 164
tern::control_riskdiff(), 44, 49
tern::coxph_pairwise(), 173
tern::estimate_odds_ratio(), 95
tern::estimate_proportion_diff(), 95, 96
tern::h_stack_by_baskets(), 160
tern::summarize_glm_count(), 128
tern::surv_time(), 173
tern::surv_timepoint(), 172, 173
tern::test_proportion_diff(), 95, 96
tm_a_gee, 24
tm_a_mmrm, 28
tm_g_barchart_simple, 33
tm_g_ci, 38
tm_g_ci(), 41
tm_g_forest_rsp, 42
tm_g_forest_tte, 48
tm_g_ipp, 52
tm_g_km, 57
tm_g_lineplot, 62
tm_g_pp_adverse_events, 66
tm_g_pp_patient_timeline, 70
tm_g_pp_therapy, 74
tm_g_pp_vitals, 78
tm_t_abnormality, 81
tm_t_abnormality_by_worst_grade, 86
tm_t_ancova, 89
tm_t_binary_outcome, 94
tm_t_coxreg, 99
tm_t_events, 105
tm_t_events_by_grade, 109
tm_t_events_patyear, 113
tm_t_events_summary, 117
tm_t_exposure, 122
tm_t_glm_counts, 126
tm_t_logistic, 130
tm_t_mult_events, 134
tm_t_pp_basic_info, 138
tm_t_pp_laboratory, 140
tm_t_pp_medical_history, 142
tm_t_pp_prior_medication, 145
tm_t_shift_by_arm, 147
tm_t_shift_by_arm_by_worst, 151
tm_t_shift_by_grade, 155
tm_t_smq, 159
tm_t_summary, 162
tm_t_summary_by, 166
tm_t_tte, 170
tmc_ex_adae (ex_data), 15
tmc_ex_adaette (ex_data), 15
tmc_ex_adcm (ex_data), 15
tmc_ex_adeg (ex_data), 15
tmc_ex_adex (ex_data), 15
tmc_ex_adlb (ex_data), 15
tmc_ex_admh (ex_data), 15
tmc_ex_adqs (ex_data), 15
tmc_ex_adrs (ex_data), 15
tmc_ex_adsl (ex_data), 15
tmc_ex_adtte (ex_data), 15
tmc_ex_advs (ex_data), 15