

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

September 25, 2024

**Title** 'teal' Modules for Standard Clinical Outputs

**Version** 0.9.1

**Date** 2024-04-26

**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 (>= 4.0), teal (>= 0.15.2), teal.transform (>= 0.5.0), tern (>= 0.9.4)

**Imports** broom (>= 0.7.10), checkmate (>= 2.1.0), cowplot (>= 0.7.0), dplyr (>= 1.0.5), DT (>= 0.13), ggplot2 (>= 3.4.0), ggrepel, grid, lifecycle (>= 0.2.0), methods, rlistings (>= 0.2.8), rmarkdown (>= 2.19), rtables (>= 0.6.7), scales, shiny (>= 1.6.0), shinyjs, shinyvalidate, shinyWidgets (>= 0.5.1), stats, teal.code (>= 0.5.0), teal.data (>= 0.4.0), teal.logger (>= 0.2.0), teal.reporter (>= 0.2.1), teal.widgets (>= 0.4.0), tern.gee (>= 0.1.3), tern.mmrm (>= 0.3.0), utils, vstime (>= 1.2.3)

**Suggests** forcats, knitr (>= 1.34), logger (>= 0.2.0), lubridate (>= 1.7.9), nestcolor (>= 0.1.0), pkgload, shinytest2, styler, testthat (>= 3.1.5), withr (> 2.0.1.5)

**VignetteBuilder** knitr

**Config/Needs/verdepcheck** insightsengineering/teal, insightsengineering/teal.transform, insightsengineering/tern, tidymodels/broom, mllg/checkmate, 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.mmrn, tidyverse/tidyr,  
 shosaco/vistime, tidyverse/forcats, yihui/knitr,  
 tidyverse/lubridate, insightsengineering/nestcolor,  
 r-lib/pkgload, rstudio/shinytest2, r-lib/styler,  
 r-lib/testthat, r-lib/withr

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

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

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

**RemoteUrl** <https://github.com/insightsengineering/teal.modules.clinical>

**RemoteRef** v0.9.1

**RemoteSha** 73902e47b6a15fb983c21b3811e6e255de9b5388

## 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
extract_input . . . . .	15
ex_data . . . . .	15
get_var_labels . . . . .	17
h_concat_expr . . . . .	18

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_mmr . . . . .	27
tm_g_barchart_simple . . . . .	30
tm_g_ci . . . . .	34
tm_g_forest_rsp . . . . .	37
tm_g_forest_tte . . . . .	41
tm_g_ipp . . . . .	45
tm_g_km . . . . .	48
tm_g_lineplot . . . . .	52
tm_g_pp_adverse_events . . . . .	56
tm_g_pp_patient_timeline . . . . .	59
tm_g_pp_therapy . . . . .	63
tm_g_pp_vitals . . . . .	66
tm_t_abnormality . . . . .	69
tm_t_abnormality_by_worst_grade . . . . .	72
tm_t_ancova . . . . .	75
tm_t_binary_outcome . . . . .	79
tm_t_coxreg . . . . .	83
tm_t_events . . . . .	88
tm_t_events_by_grade . . . . .	91
tm_t_events_patyear . . . . .	94
tm_t_events_summary . . . . .	97
tm_t_exposure . . . . .	101
tm_t_logistic . . . . .	104
tm_t_mult_events . . . . .	107
tm_t_pp_basic_info . . . . .	110
tm_t_pp_laboratory . . . . .	111
tm_t_pp_medical_history . . . . .	114
tm_t_pp_prior_medication . . . . .	116
tm_t_shift_by_arm . . . . .	118
tm_t_shift_by_arm_by_worst . . . . .	121
tm_t_shift_by_grade . . . . .	124
tm_t_smq . . . . .	127
tm_t_summary . . . . .	131
tm_t_summary_by . . . . .	133
tm_t_tte . . . . .	137

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:** Joe Zhu <joe.zhu@roche.com>

Authors:

- 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$rsp_lab <- d_onco_rsp_label(anl$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	<i>Concatenate expressions via a binary operator</i>
------------------	--

---

## 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	<i>Clean up categorical variable description</i>
-------------------	--

---

**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	<i>Mapping function for Laboratory Table</i>
------------------	--

---

**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	<i>Convert choices_selected to data_extract_spec with only filter_spec</i>
------------------	--

---

**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 <a href="#">teal.transform::choices_selected()</a> for details.
dataname	(character) name of the data
multiple	(logical) Whether multiple values shall be allowed in the shiny <a href="#">shiny::selectInput()</a> .
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	<i>Convert choices_selected to data_extract_spec with only select_spec</i>
------------------	--

---

### 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 <a href="#">teal.transform::choices_selected()</a> for details.
dataname	(character) name of the data
multiple	(logical) Whether multiple values shall be allowed in the shiny <a href="#">shiny::selectInput()</a> .
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	<i>Convert choices_selected to filter_spec</i>
-------------------	--

---

### 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 <a href="#">teal.transform::choices_selected()</a> for details.
multiple	(logical) Whether multiple values shall be allowed in the shiny <a href="#">shiny::selectInput()</a> .
label	(character) Label to print over the selection field. For no label, set to NULL.

**Value**

([teal.transform::filter\\_spec\(\)](#))

---

cs_to_select_spec	<i>Convert choices_selected to select_spec</i>
-------------------	--

---

**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 <a href="#">teal.transform::choices_selected()</a> for details.
multiple	(logical) Whether multiple values shall be allowed in the shiny <a href="#">shiny::selectInput()</a> .
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	<i>Default string for total column label</i>
---------------------	--

---

## Description

### [Stable]

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

<code>total_label</code>	(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")
```

---

extract_input	<i>Extracts html id for data_extract_ui</i>
---------------	---

---

**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]]\$dataname
- filter (logical) optional,  
if the connected extract\_data\_spec has objects passed to its filter argument

**Value**

a string

**Examples**

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

---

ex_data	<i>Simulated CDISC Data for Examples</i>
---------	--

---

**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\_advsv

**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\_adeq: 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\_advts: ADVS data

---

`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	<i>Expression Deparsing</i>
---------------	-----------------------------

---

**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	<i>Whether object is of class <code>teal.transform::choices_selected()</code></i>
--------------	---

---

**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	<i>Expression: Prepare Arm Levels</i>
--------------------	---------------------------------------

---

**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	<i>Split choices_selected objects with interactions into their component variables</i>
---------------	--

---

**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 `choices_selected()` object.

**Note**

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

**Examples**

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

---

split_col_expr	<i>Split-Column Expression</i>
----------------	--------------------------------

---

**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	<i>Split interaction terms into their component variables</i>
--------------------	---

---

**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	<i>teal Module: Generalized Estimating Equations (GEE) analysis</i>
----------	---

---

**Description**

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

**Usage**

```
tm_a_gee(
  label,
  dataname,
  parentname = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  aval_var,
  id_var,
  arm_var,
  visit_var,
```



```

    cov_var,
    arm_ref_comp = NULL,
    paramcd,
    conf_level = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
      TRUE),
    pre_output = NULL,
    post_output = NULL,
    basic_table_args = teal.widgets::basic_table_args()
  )

```

## 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.transform::choices_selected()) object with all available choices and pre-selected option for the analysis variable.
id_var	(teal.transform::choices_selected()) object specifying the variable name for subject id.
arm_var	(teal.transform::choices_selected()) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
visit_var	(teal.transform::choices_selected()) 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.transform::choices_selected()) 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.transform::variable_choices() or delayed teal.transform::value_choices() with the elements named ref and comp that the defined the default reference and comparison arms when the arm variable is changed.
paramcd	(teal.transform::choices_selected()) object with all available choices and preselected option for the parameter code variable from dataname.
conf_level	(teal.transform::choices_selected()) object with all available choices and pre-selected option for the confidence level, each within range of (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")`.

## Value

a `teal_module` object.

## 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
  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()
})
datanames <- c("ADSL", "ADQS")
datanames(data) <- datanames
join_keys(data) <- default_cdisc_join_keys[datanames]

app <- init(
  data = data,
  modules = modules(
    tm_a_gee(
      label = "GEE",
      dataname = "ADQS",
      aval_var = choices_selected("AVALBIN", fixed = TRUE),
```

```

    id_var = choices_selected(c("USUBJID", "SUBJID"), "USUBJID"),
    arm_var = choices_selected(c("ARM", "ARMCD"), "ARM"),
    visit_var = choices_selected(c("AVISIT", "AVISITN"), "AVISIT"),
    paramcd = choices_selected(
      choices = value_choices(data[["ADQS"]], "PARAMCD", "PARAM"),
      selected = "FKSI-FWB"
    ),
    cov_var = choices_selected(c("BASE", "AGE", "SEX", "BASE:AVISIT"), NULL)
  )
}
)
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}

```

tm\_a\_mmrn

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

## Description

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

## Usage

```

tm_a_mmrn(
  label,
  dataname,
  parentname = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  aval_var,
  id_var,
  arm_var,
  visit_var,
  cov_var,
  arm_ref_comp = NULL,
  paramcd,
  method = teal.transform::choices_selected(c("Satterthwaite", "Kenward-Roger",
    "Kenward-Roger-Linear"), "Satterthwaite", keep_order = TRUE),
  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,
  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()
)

```

**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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
id_var	( <a href="#">teal.transform::choices_selected()</a> ) object specifying the variable name for subject id.
arm_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
visit_var	( <a href="#">teal.transform::choices_selected()</a> ) 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	( <a href="#">teal.transform::choices_selected()</a> ) 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 <a href="#">teal.transform::variable_choices()</a> or delayed <a href="#">teal.transform::value_choices()</a> with the elements named ref and comp that the defined the default reference and comparison arms when the arm variable is changed.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
method	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the adjustment method.
conf_level	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the confidence level, each within range of (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 <code>set_default_total_label("new_default")</code> .
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 <code>shiny::helpText()</code> elements are useful.
basic_table_args	(basic_table_args) optional object created by <code>teal.widgets::basic_table_args()</code> with settings for the module table. The argument is merged with option <code>teal.basic_table_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-basic-table-arguments", package = "teal.widgets")</code> .
ggplot2_args	(ggplot2_args) optional object created by <code>teal.widgets::ggplot2_args()</code> with settings for all the plots or named list of ggplot2_args objects for plot-specific settings. List names should match the following: <code>c("default", "lsmeans", "diagnostic")</code> . The argument is merged with option <code>teal.ggplot2_args</code> and with default module arguments (hard coded in the module body). For more details, see the help vignette: <code>vignette("custom-ggplot2-arguments", package = "teal.widgets")</code> .

**Value**

a teal\_module object.

**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**

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

data <- teal_data()
data <- within(data, {
```

```

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
  )
})

datanames <- c("ADSL", "ADQS")
datanames(data) <- datanames
join_keys(data) <- default_cdisc_join_keys[datanames]
app <- init(
  data = data,
  modules = modules(
    tm_a_mmrn(
      label = "MMRM",
      dataname = "ADQS",
      aval_var = choices_selected(c("AVAL", "CHG"), "AVAL"),
      id_var = choices_selected(c("USUBJID", "SUBJID"), "USUBJID"),
      arm_var = choices_selected(c("ARM", "ARMCD"), "ARM"),
      visit_var = choices_selected(c("AVISIT", "AVISITN"), "AVISIT"),
      arm_ref_comp = arm_ref_comp,
      paramcd = choices_selected(
        choices = value_choices(data[["ADQS"]], "PARAMCD", "PARAM"),
        selected = "FKSI-FWB"
      ),
      cov_var = choices_selected(c("BASE", "AGE", "SEX", "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()
)
```

**Arguments**

x	(data_extract_spec) variable on the x-axis.
fill	(data_extract_spec) grouping variable to determine bar colors.
x_facet	(data_extract_spec) row-wise faceting groups.
y_facet	(data_extract_spec) column-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 <a href="#">shiny::helpText()</a> elements are useful.
ggplot2_args	(ggplot2_args) optional object created by <a href="#">teal.widgets::ggplot2_args()</a> 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")`.

## Details

Categories can be defined up to four levels deep and are defined through the `x`, `fill`, `x_facet`, and `y_facet` parameters. Any parameters set to `NULL` (default) are ignored.

## Value

a `teal_module` object.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
library(nestcolor)
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")))

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADAE = ADAE,
    code = "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\")))"
  ),
  modules = modules(
    tm_g_barchart_simple(
      label = "ADAE Analysis",
      x = data_extract_spec(
        dataname = "ADSL",
        select = select_spec(
          choices = variable_choices(
            ADSL,
            c(
              "ARM", "ACTARM", "SEX",
              "RACE", "ITTFL", "SAFFL", "STRATA2"
            )
          ),
          selected = "ACTARM",
```



```

        multiple = FALSE
      )
    ),
    fill = list(
      data_extract_spec(
        dataname = "ADSL",
        select = select_spec(
          choices = variable_choices(
            ADSL,
            c(
              "ARM", "ACTARM", "SEX",
              "RACE", "ITTFL", "SAFFL", "STRATA2"
            )
          ),
          selected = "SEX",
          multiple = FALSE
        )
      ),
      data_extract_spec(
        dataname = "ADAE",
        select = select_spec(
          choices = variable_choices(ADAE, c("AETOXGR", "AESEV", "AESER")),
          selected = NULL,
          multiple = FALSE
        )
      )
    ),
    x_facet = list(
      data_extract_spec(
        dataname = "ADAE",
        select = select_spec(
          choices = variable_choices(ADAE, c("AETOXGR", "AESEV", "AESER")),
          selected = "AETOXGR",
          multiple = FALSE
        )
      ),
      data_extract_spec(
        dataname = "ADSL",
        select = select_spec(
          choices = variable_choices(
            ADSL,
            c(
              "ARM", "ACTARM", "SEX",
              "RACE", "ITTFL", "SAFFL", "STRATA2"
            )
          ),
          selected = NULL,
          multiple = FALSE
        )
      )
    ),
    y_facet = list(
      data_extract_spec(

```

```

    dataname = "ADAE",
    select = select_spec(
      choices = variable_choices(ADAE, c("AETOXGR", "AESEV", "AESER")),
      selected = "AESEV",
      multiple = FALSE
    )
  ),
  data_extract_spec(
    dataname = "ADSL",
    select = select_spec(
      choices = variable_choices(
        ADSL,
        c(
          "ARM", "ACTARM", "SEX",
          "RACE", "ITTFL", "SAFFL", "STRATA2"
        )
      ),
      selected = NULL,
      multiple = FALSE
    )
  )
)
)
)
)
)
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}

```

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,
  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()
  )

```

## Arguments

label	(character) menu item label of the module in the teal app.
x_var	(character) name of the treatment variable to put on the x-axis.
y_var	(character) name of the response variable to put on the y-axis.
color	(data_extract_spec) the group variable used to determine the plot colors, shapes, and line types.
stat	(character) statistic to plot. Options are "mean" and "median".
conf_level	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the confidence level, each within range of (0, 1).
plot_height	(numeric) optional vector of length three with <code>c(value, min, max)</code> . 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 <code>c(value, min, max)</code> . 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 <a href="#">shiny::helpText()</a> elements are useful.
ggplot2_args	(ggplot2_args) optional object created by <a href="#">teal.widgets::ggplot2_args()</a> with settings for the module plot. The argument is merged with option <code>teal.ggplot2_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-ggplot2-arguments", package = "teal.widgets")</code> .

## Value

a teal\_module object.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```

library(nestcolor)

ADSL <- tmc_ex_ads1
ADLB <- tmc_ex_adlb

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADLB = ADLB,
    code = "
      ADSL <- tmc_ex_ads1
      ADLB <- tmc_ex_adlb
    "
  ),
  modules = modules(
    tm_g_ci(
      label = "Confidence Interval Plot",
      x_var = data_extract_spec(
        dataname = "ADSL",
        select = select_spec(
          choices = c("ARMCD", "BMRKR2"),
          selected = c("ARMCD"),
          multiple = FALSE,
          fixed = FALSE
        )
      ),
      y_var = data_extract_spec(
        dataname = "ADLB",
        filter = list(
          filter_spec(
            vars = "PARAMCD",
            choices = levels(ADLB$PARAMCD),
            selected = levels(ADLB$PARAMCD)[1],
            multiple = FALSE,
            label = "Select lab:"
          ),
          filter_spec(
            vars = "AVISIT",
            choices = levels(ADLB$AVISIT),
            selected = levels(ADLB$AVISIT)[1],
            multiple = FALSE,
            label = "Select visit:"
          )
        ),
      select = select_spec(
        label = "Analyzed Value",
        choices = c("AVAL", "CHG"),
        selected = "AVAL",
        multiple = FALSE,
        fixed = FALSE
      )
    )
  )

```

```

    ),
    color = data_extract_spec(
      dataname = "ADSL",
      select = select_spec(
        label = "Color by variable",
        choices = c("SEX", "STRATA1", "STRATA2"),
        selected = c("STRATA1"),
        multiple = FALSE,
        fixed = FALSE
      )
    )
  ),
  header = "Example of Confidence Interval Plot",
  footer = tags$p(
    class = "text-muted", "Source: `teal.modules.clinical::tm_g_ci`"
  )
)
if (interactive()) {
  shinyApp(app$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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  arm_ref_comp = NULL,
  paramcd,
  aval_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AVALC"), "AVALC", fixed = TRUE),
  subgroup_var,
  strata_var,
  fixed_symbol_size = TRUE,
  conf_level = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
    TRUE),
  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()
)

```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
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 <a href="#">teal.transform::variable_choices()</a> or delayed <a href="#">teal.transform::value_choices()</a> with the elements named ref and comp that the defined the default reference and comparison arms when the arm variable is changed.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
aval_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
subgroup_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as the default subgroups.
strata_var	( <a href="#">teal.transform::choices_selected()</a> ) names of the variables for stratified analysis.
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 subgroup.
conf_level	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the confidence level, each within range of (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 <a href="#">shiny::helpText()</a> elements are useful.
ggplot2_args	(ggplot2_args) optional object created by <a href="#">teal.widgets::ggplot2_args()</a> 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: <code>vignette("custom-ggplot2-arguments", package = "teal.widgets")</code> .

**Value**

a teal\_module object.

**See Also**

The [TLG Catalog](#) where additional example apps implementing this module can be found.

**Examples**

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

```

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")

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 = cdisc_data(
    ADSL = ADSL,
    ADRS = ADRS,
    code = "
      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\")
    "
  ),
  modules = modules(
    tm_g_forest_rsp(
      label = "Forest Response",
      dataname = "ADRS",
      arm_var = choices_selected(
        variable_choices(ADSL, c("ARM", "ARMCD")),
        "ARMCD"
      ),
      arm_ref_comp = arm_ref_comp,
      paramcd = choices_selected(
        value_choices(ADRS, "PARAMCD", "PARAM"),
        "INVET"
      ),
      subgroup_var = choices_selected(
        variable_choices(ADSL, names(ADSL)),
        c("BMRKR2", "SEX")
      ),
      strata_var = choices_selected(
        variable_choices(ADSL, c("STRATA1", "STRATA2")),
        "STRATA2"
      ),
      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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  arm_ref_comp = NULL,
  subgroup_var,
  paramcd,
  strata_var,
  aval_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AVAL"), "AVAL", fixed = TRUE),
  cnsr_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,

```

```

"CNSR"), "CNSR", fixed = TRUE),
conf_level = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
  TRUE),
time_unit_var =
  teal.transform::choices_selected(teal.transform::variable_choices(dataname, "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()
)

```

## 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.transform::choices_selected()) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
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.transform::variable_choices() or delayed teal.transform::value_choices() with the elements named ref and comp that the defined the default reference and comparison arms when the arm variable is changed.
subgroup_var	(teal.transform::choices_selected()) object with all available choices and preselected option for variable names that can be used as the default subgroups.
paramcd	(teal.transform::choices_selected()) object with all available choices and preselected option for the parameter code variable from dataname.
strata_var	(teal.transform::choices_selected()) names of the variables for stratified analysis.
aval_var	(teal.transform::choices_selected()) object with all available choices and pre-selected option for the analysis variable.

cnsr_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the censoring variable.
conf_level	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the confidence level, each within range of (0, 1).
time_unit_var	( <a href="#">teal.transform::choices_selected()</a> ) 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 subgroup.
plot_height	(numeric) optional vector of length three with <code>c(value, min, max)</code> . 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 <code>c(value, min, max)</code> . 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 <code>1 - rel_width_forest</code> . If <code>as_list = TRUE</code> , 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 <a href="#">shiny::helpText()</a> elements are useful.
ggplot2_args	(ggplot2_args) optional object created by <a href="#">teal.widgets::ggplot2_args()</a> with settings for the module plot. The argument is merged with option <code>teal.ggplot2_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-ggplot2-arguments", package = "teal.widgets")</code> .

**Value**

a `teal_module` object.

**See Also**

The [TLG Catalog](#) where additional example apps implementing this module can be found.

**Examples**

```

library(nestcolor)

ADSL <- tmc_ex_adsl
ADTTE <- tmc_ex_adtte
ADSL$RACE <- droplevels(ADSL$RACE) %>% with_label("Race")

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 = cdisc_data(
    ADSL = ADSL,
    ADTTE = ADTTE,
    code = "
      ADSL <- tmc_ex_adsl
      ADTTE <- tmc_ex_adtte
      ADSL$RACE <- droplevels(ADSL$RACE) %>% with_label(\"Race\")
    "
  ),
  modules = modules(
    tm_g_forest_tte(
      label = "Forest Survival",
      dataname = "ADTTE",
      arm_var = choices_selected(
        variable_choices(ADSL, c("ARM", "ARMCD")),
        "ARMCD"
      ),
      arm_ref_comp = arm_ref_comp,
      paramcd = choices_selected(
        value_choices(ADTTE, "PARAMCD", "PARAM"),
        "OS"
      ),
      subgroup_var = choices_selected(
        variable_choices(ADSL, names(ADSL)),
        c("BMRKR2", "SEX")
      ),
      strata_var = choices_selected(
        variable_choices(ADSL, c("STRATA1", "STRATA2")),
        "STRATA2"
      )
    )
  )
)

```

```

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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  paramcd,
  id_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "USUBJID"), "USUBJID", fixed = TRUE),
  visit_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AVISIT"), "AVISIT", fixed = TRUE),
  aval_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AVAL"), "AVAL", fixed = TRUE),
  avalu_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AVALU"), "AVALU", fixed = TRUE),
  base_var = lifecycle::deprecated(),
  baseline_var =
    teal.transform::choices_selected(teal.transform::variable_choices(dataname, "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()
)

```

**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	( <code>teal.transform::choices_selected()</code> ) object with all available choices and preselected option for variable values that can be used as arm variable.
paramcd	( <code>teal.transform::choices_selected()</code> ) object with all available choices and preselected option for the parameter code variable from dataname.
id_var	( <code>teal.transform::choices_selected()</code> ) object specifying the variable name for subject id.
visit_var	( <code>teal.transform::choices_selected()</code> ) 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	( <code>teal.transform::choices_selected()</code> ) object with all available choices and pre-selected option for the analysis variable.
avalu_var	( <code>teal.transform::choices_selected()</code> ) object with all available choices and preselected option for the analysis unit variable.
base_var	<b>[Deprecated]</b> Please use the baseline_var argument instead.
baseline_var	( <code>teal.transform::choices_selected()</code> ) 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 <code>c(value, min, max)</code> . 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 <code>c(value, min, max)</code> . 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")`.

### Value

a teal\_module object.

### See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

### Examples

```
library(nestcolor)
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")

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADLB = ADLB,
    code = "
      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\")
    "
  ),
  modules = modules(
    tm_g_ipp(
      label = "Individual Patient Plot",
```

```

    dataname = "ADLB",
    arm_var = choices_selected(
      value_choices(ADLB, "ARMCD"),
      "ARM A"
    ),
    paramcd = choices_selected(
      value_choices(ADLB, "PARAMCD"),
      "ALT"
    ),
    aval_var = choices_selected(
      variable_choices(ADLB, c("AVAL", "CHG")),
      "AVAL"
    ),
    avalu_var = choices_selected(
      variable_choices(ADLB, c("AVALU")),
      "AVALU",
      fixed = TRUE
    ),
    id_var = choices_selected(
      variable_choices(ADLB, c("USUBJID")),
      "USUBJID",
      fixed = TRUE
    ),
    visit_var = choices_selected(
      variable_choices(ADLB, c("AVISIT")),
      "AVISIT"
    ),
    baseline_var = choices_selected(
      variable_choices(ADLB, c("BASE")),
      "BASE",
      fixed = TRUE
    ),
    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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  arm_ref_comp = NULL,
  paramcd,
  strata_var,
  facet_var,
  time_unit_var =
    teal.transform::choices_selected(teal.transform::variable_choices(dataname, "AVALU"),
      "AVALU", fixed = TRUE),
  aval_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AVAL"), "AVAL", fixed = TRUE),
  cnsr_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "CNSR"), "CNSR", fixed = TRUE),
  conf_level = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
    TRUE),
  font_size = c(11L, 1L, 30),
  control_annot_surv_med = control_surv_med_annot(),
  control_annot_coxph = 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
)

```

**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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
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

	<p><a href="#">teal.transform::variable_choices()</a> or delayed <a href="#">teal.transform::value_choices()</a> with the elements named <code>ref</code> and <code>comp</code> that the defined the default reference and comparison arms when the arm variable is changed.</p>
<code>paramcd</code>	<p>(<a href="#">teal.transform::choices_selected()</a>) object with all available choices and preselected option for the parameter code variable from <code>dataname</code>.</p>
<code>strata_var</code>	<p>(<a href="#">teal.transform::choices_selected()</a>) names of the variables for stratified analysis.</p>
<code>facet_var</code>	<p>(<a href="#">teal.transform::choices_selected()</a>) object with all available choices and preselected option for names of variable that can be used for plot faceting.</p>
<code>time_unit_var</code>	<p>(<a href="#">teal.transform::choices_selected()</a>) object with all available choices and pre-selected option for the time unit variable.</p>
<code>aval_var</code>	<p>(<a href="#">teal.transform::choices_selected()</a>) object with all available choices and pre-selected option for the analysis variable.</p>
<code>cnsr_var</code>	<p>(<a href="#">teal.transform::choices_selected()</a>) object with all available choices and preselected option for the censoring variable.</p>
<code>conf_level</code>	<p>(<a href="#">teal.transform::choices_selected()</a>) object with all available choices and pre-selected option for the confidence level, each within range of (0, 1).</p>
<code>font_size</code>	<p>(numeric) numeric vector of length 3 of current, minimum and maximum font size values.</p>
<code>control_annot_surv_med</code>	<p>(list) parameters to control the position and size of the annotation table added to the plot when <code>annot_surv_med = TRUE</code>, specified using the <a href="#">control_surv_med_annot()</a> function. Parameter options are: <code>x</code>, <code>y</code>, <code>w</code>, <code>h</code>, and <code>fill</code>. See <a href="#">control_surv_med_annot()</a> for details.</p>
<code>control_annot_coxph</code>	<p>(list) parameters to control the position and size of the annotation table added to the plot when <code>annot_coxph = TRUE</code>, specified using the <a href="#">control_coxph_annot()</a> function. Parameter options are: <code>x</code>, <code>y</code>, <code>w</code>, <code>h</code>, <code>fill</code>, and <code>ref_lbls</code>. See <a href="#">control_coxph_annot()</a> for details.</p>
<code>legend_pos</code>	<p>(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.</p>
<code>rel_height_plot</code>	<p>(proportion) proportion of total figure height to allocate to the Kaplan-Meier plot. Relative height of patients at risk table is then <math>1 - \text{rel\_height\_plot}</math>. If <code>annot_at_risk = FALSE</code> or <code>as_list = TRUE</code>, this parameter is ignored.</p>

plot_height	(numeric) optional vector of length three with <code>c(value, min, max)</code> . 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 <code>c(value, min, max)</code> . 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 <code>shiny::helpText()</code> elements are useful.

### Value

a teal\_module object.

### See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

### Examples

```
library(nestcolor)

ADSL <- tmc_ex_adsl
ADTTE <- tmc_ex_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 = cdisc_data(
    ADSL = ADSL,
    ADTTE = ADTTE,
    code = "
      ADSL <- tmc_ex_adsl
      ADTTE <- tmc_ex_adtte
    "
  ),
  modules = modules(
    tm_g_km(
      label = "Kaplan-Meier Plot",
```

```

    dataname = "ADTTE",
    arm_var = choices_selected(
      variable_choices(ADSL, c("ARM", "ARMCD", "ACTARMCD")),
      "ARM"
    ),
    paramcd = choices_selected(
      value_choices(ADTTE, "PARAMCD", "PARAM"),
      "OS"
    ),
    arm_ref_comp = arm_ref_comp,
    strata_var = choices_selected(
      variable_choices(ADSL, c("SEX", "BMRKR2")),
      "SEX"
    ),
    facet_var = choices_selected(
      variable_choices(ADSL, c("SEX", "BMRKR2")),
      NULL
    )
  )
)
)
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 = ifelse(inherits(strata, "data_extract_spec"),
    teal.transform::datanames_input(strata, "ADSL"),
    strata = teal.transform::choices_selected(teal.transform::variable_choices(parentname,
      c("ARM", "ARMCD", "ACTARMCD")), "ARM"),
  x = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AVISIT"), "AVISIT", fixed = TRUE),
  y = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    c("AVAL", "BASE", "CHG", "PCHG")), "AVAL"),
  y_unit = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AVALU"), "AVALU", fixed = TRUE),

```

```

paramcd = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
  "PARAMCD"), "PARAMCD", fixed = TRUE),
param = teal.transform::choices_selected(teal.transform::value_choices(dataname,
  "PARAMCD", "PARAM"), "ALT"),
conf_level = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
  TRUE),
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()
)

```

### 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	<b>[Deprecated]</b> use the group_var parameter instead.
x	(string) x-variable name.
y	(string) y-variable name.
y_unit	(string or NA) y-axis unit variable name.
paramcd	(teal.transform::choices_selected()) object with all available choices and preselected option for the parameter code variable from dataname.
param	(character) parameter to filter the data by.
conf_level	(teal.transform::choices_selected()) object with all available choices and pre-selected option for the confidence level, each within range of (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 <code>interval = NULL</code> if intervals should not be added to the plot.
<code>mid</code>	(character or NULL) names of the statistics that will be plotted as midpoints. All the statistics indicated in <code>mid</code> variable must be present in the object returned by <code>sfun</code> , and be of a double or numeric type vector of length one.
<code>whiskers</code>	(character) names of the interval whiskers that will be plotted. Names must match names of the list element <code>interval</code> that will be returned by <code>sfun</code> (e.g. <code>mean_ci_lwr</code> element of <code>sfun(x)[["mean_ci"]]</code> ). It is possible to specify one whisker only, or to suppress all whiskers by setting <code>interval = NULL</code> .
<code>table</code>	(character or NULL) names of the statistics that will be displayed in the table below the plot. All the statistics indicated in <code>table</code> variable must be present in the object returned by <code>sfun</code> .
<code>mid_type</code>	(string) controls the type of the mid plot, it can be point (" <code>p</code> "), line (" <code>l</code> "), or point and line (" <code>pl</code> ").
<code>mid_point_size</code>	(numeric(1)) font size of the mid plot points.
<code>table_font_size</code>	(numeric(1)) font size of the text in the table.
<code>plot_height</code>	(numeric) optional vector of length three with <code>c(value, min, max)</code> . Specifies the height of the main plot and renders a slider on the plot to interactively adjust the plot height.
<code>plot_width</code>	(numeric) optional vector of length three with <code>c(value, min, max)</code> . Specifies the width of the main plot and renders a slider on the plot to interactively adjust the plot width.
<code>pre_output</code>	(shiny.tag) optional, with text placed before the output to put the output into context. For example a title.
<code>post_output</code>	(shiny.tag) optional, with text placed after the output to put the output into context. For example the <a href="#">shiny::helpText()</a> elements are useful.
<code>ggplot2_args</code>	(ggplot2_args) optional object created by <code>teal.widgets::ggplot2_args()</code> with settings for the module plot. For this module, this argument will only accept <code>ggplot2_args</code> object with <code>labs</code> list of following child elements: <code>title</code> , <code>subtitle</code> , <code>caption</code> , <code>y</code> , <code>lty</code> . No other elements would be taken into account. The argument is merged with option <code>teal.ggplot2_args</code> and with default module arguments (hard coded in the module body).  For more details, see the vignette: <code>vignette("custom-ggplot2-arguments", package = "teal.widgets")</code> .

**Value**

a teal\_module object.

**See Also**

The [TLG Catalog](#) where additional example apps implementing this module can be found.

**Examples**

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

ADSL <- tmc_ex_adsl
ADLB <- tmc_ex_adlb %>% mutate(AVISIT == fct_reorder(AVISIT, AVISITN, min))

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADLB = ADLB,
    code = "
      ADSL <- tmc_ex_adsl
      ADLB <- tmc_ex_adlb %>% mutate(AVISIT == fct_reorder(AVISIT, AVISITN, min))
    "
  ),
  modules = modules(
    tm_g_lineplot(
      label = "Line Plot",
      dataname = "ADLB",
      strata = choices_selected(
        variable_choices(ADSL, c("ARM", "ARMCD", "ACTARMCD")),
        "ARM"
      ),
      y = choices_selected(
        variable_choices(ADLB, c("AVAL", "BASE", "CHG", "PCHG")),
        "AVAL"
      ),
      param = choices_selected(
        value_choices(ADLB, "PARAMCD", "PARAM"),
        "ALT"
      )
    )
  )
)
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()
)
```

## 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	( <code>teal.transform::choices_selected()</code> ) object with all available choices and preselected option for the AETERM variable from dataname.



tox_grade	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the AETOXGR variable from dataname.
causality	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the AEREL variable from dataname.
outcome	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the AEOUT variable from dataname.
action	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the AEACN variable from dataname.
time	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the ASTDY variable from dataname.
decod	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>c(value, min, max)</code> . 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 <code>c(value, min, max)</code> . Specifies the width of the main plot and renders a slider on the plot to interactively adjust the plot width.
pre_output	( <code>shiny.tag</code> ) optional, with text placed before the output to put the output into context. For example a title.
post_output	( <code>shiny.tag</code> ) optional, with text placed after the output to put the output into context. For example the <a href="#">shiny::helpText()</a> elements are useful.
ggplot2_args	( <code>ggplot2_args</code> ) optional object created by <a href="#">teal.widgets::ggplot2_args()</a> with settings for the module plot. The argument is merged with option <code>teal.ggplot2_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-ggplot2-arguments", package = "teal.widgets")</code> .

## Value

a `teal_module` object.

**Examples**

```

library(nestcolor)
library(dplyr)

ADAE <- tmc_ex_adae
ADSL <- tmc_ex_adsl %>% filter(USUBJID %in% ADAE$USUBJID)

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADAE = ADAE,
    code = "
      ADAE <- tmc_ex_adae
      ADSL <- tmc_ex_adsl %>% filter(USUBJID %in% ADAE$USUBJID)
    "
  ),
  modules = modules(
    tm_g_pp_adverse_events(
      label = "Adverse Events",
      dataname = "ADAE",
      parentname = "ADSL",
      patient_col = "USUBJID",
      plot_height = c(600L, 200L, 2000L),
      aeterm = choices_selected(
        choices = variable_choices(ADAE, "AETERM"),
        selected = "AETERM"
      ),
    ),
    tox_grade = choices_selected(
      choices = variable_choices(ADAE, "AETOXGR"),
      selected = "AETOXGR"
    ),
    ),
    causality = choices_selected(
      choices = variable_choices(ADAE, "AEREL"),
      selected = "AEREL"
    ),
    ),
    outcome = choices_selected(
      choices = variable_choices(ADAE, "AEOUT"),
      selected = "AEOUT"
    ),
    ),
    action = choices_selected(
      choices = variable_choices(ADAE, "AEACN"),
      selected = "AEACN"
    ),
    ),
    time = choices_selected(
      choices = variable_choices(ADAE, "ASTDY"),
      selected = "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()
)

```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the AETERM variable from dataname.
cmdecod	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMDECOD variable from dataname_adcm.
aetime_start	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the ASTDTM variable from dataname_adae.
aetime_end	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the AENDTM variable from dataname_adae.
dstime_start	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMASTDTM variable from dataname_adcm.
dstime_end	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMAENDTM variable from dataname_adcm.
aerelday_start	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the ASTDY variable from dataname_adae.
aerelday_end	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the AENDY variable from dataname_adae.
dsrelday_start	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the ASTDY variable from dataname_adcm.
dsrelday_end	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>c(value, min, max)</code> . 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 <code>c(value, min, max)</code> . Specifies the width of the main plot and renders a slider on the plot to interactively adjust the plot width.
pre_output	( <code>shiny.tag</code> ) 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")`.

## Value

a `teal_module` object.

## Examples

```
library(nestcolor)
library(dplyr)

data <- teal_data()
data <- within(data, {
  ADAE <- tmc_ex_adae
  ADSL <- tmc_ex_adsl %>% filter(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
  )
})

adcm_keys <- c("STUDYID", "USUBJID", "ASTDTM", "CMSEQ", "ATC1", "ATC2", "ATC3", "ATC4")
datanames(data) <- c("ADSL", "ADAE", "ADCM")
join_keys(data) <- default_cdisc_join_keys[c("ADSL", "ADAE", "ADCM")]
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 = choices_selected(
  choices = variable_choices(data[["ADCM"]], "CMDECOD"),
  selected = "CMDECOD",
),
aeterm = choices_selected(
  choices = variable_choices(data[["ADAE"]], "AETERM"),
  selected = c("AETERM")
),
aetime_start = choices_selected(
  choices = variable_choices(data[["ADAE"]], "ASTDTM"),
  selected = c("ASTDTM")
),
aetime_end = choices_selected(
  choices = variable_choices(data[["ADAE"]], "AENDTM"),
  selected = c("AENDTM")
),
dstime_start = choices_selected(
  choices = variable_choices(data[["ADCM"]], "CMASTDTM"),
  selected = c("CMASTDTM")
),
dstime_end = choices_selected(
  choices = variable_choices(data[["ADCM"]], "CMAENDTM"),
  selected = c("CMAENDTM")
),
aerelday_start = choices_selected(
  choices = variable_choices(data[["ADAE"]], "ASTDY"),
  selected = c("ASTDY")
),
aerelday_end = choices_selected(
  choices = variable_choices(data[["ADAE"]], "AENDY"),
  selected = c("AENDY")
),
dsrelday_start = choices_selected(
  choices = variable_choices(data[["ADCM"]], "ASTDY"),
  selected = c("ASTDY")
),
dsrelday_end = choices_selected(
  choices = variable_choices(data[["ADCM"]], "AENDY"),
  selected = c("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()
)
```

## 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	( <code>teal.transform::choices_selected()</code> ) object with all available choices and preselected option for the ATIREL variable from dataname.

cmdecod	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMDECOD variable from dataname.
cmindc	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMINDC variable from dataname.
cmdose	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMDOSE variable from dataname.
cmtrt	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMTRT variable from dataname.
cmdosu	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMDOSU variable from dataname.
cmroute	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMROUTE variable from dataname.
cmdosfrq	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMDOSFRQ variable from dataname.
cmstdy	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMSTDY variable from dataname.
cmendy	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>c(value, min, max)</code> . 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 <code>c(value, min, max)</code> . 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 <a href="#">shiny::helpText()</a> elements are useful.
ggplot2_args	(ggplot2_args) optional object created by <a href="#">teal.widgets::ggplot2_args()</a> with settings for the module plot. The argument is merged with option <code>teal.ggplot2_args</code> and with



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

## Value

a `teal_module` object.

## Examples

```
library(nestcolor)
library(dplyr)

ADCM <- tmc_ex_adcm
ADSL <- tmc_ex_adsl %>% filter(USUBJID %in% ADCM$USUBJID)
ADCM$CMASTDTM <- ADCM$ASTDTM
ADCM$CMAENDTM <- ADCM$AENDTM
adcm_keys <- c("STUDYID", "USUBJID", "ASTDTM", "CMSEQ", "ATC1", "ATC2", "ATC3", "ATC4")

join_keys <- default_cdisc_join_keys[c("ADSL", "ADCM")]
join_keys["ADCM", "ADCM"] <- adcm_keys

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADCM = ADCM,
    code = "
      ADCM <- tmc_ex_adcm
      ADSL <- tmc_ex_adsl %>% filter(USUBJID %in% ADCM$USUBJID)
      ADCM$CMASTDTM <- ADCM$ASTDTM
      ADCM$CMAENDTM <- ADCM$AENDTM
    ",
    join_keys = join_keys
  ),
  modules = modules(
    tm_g_pp_therapy(
      label = "Therapy",
      dataname = "ADCM",
      parentname = "ADSL",
      patient_col = "USUBJID",
      plot_height = c(600L, 200L, 2000L),
      atirel = choices_selected(
        choices = variable_choices(ADCM, "ATIREL"),
        selected = c("ATIREL")
      ),
      cmdecod = choices_selected(
        choices = variable_choices(ADCM, "CMDECOD"),
        selected = "CMDECOD"
      ),
      cmindc = choices_selected(
        choices = variable_choices(ADCM, "CMINDC"),
        selected = "CMINDC"
      )
    )
  )
)
```

```

    ),
    cmdose = choices_selected(
      choices = variable_choices(ADCM, "CMDOSE"),
      selected = "CMDOSE"
    ),
    cmtrt = choices_selected(
      choices = variable_choices(ADCM, "CMTRT"),
      selected = "CMTRT"
    ),
    cmdosu = choices_selected(
      choices = variable_choices(ADCM, "CMDOSU"),
      selected = c("CMDOSU")
    ),
    cmroute = choices_selected(
      choices = variable_choices(ADCM, "CMROUTE"),
      selected = "CMROUTE"
    ),
    cmdosfrq = choices_selected(
      choices = variable_choices(ADCM, "CMDOSFRQ"),
      selected = "CMDOSFRQ"
    ),
    cmstdy = choices_selected(
      choices = variable_choices(ADCM, "ASTDY"),
      selected = "ASTDY"
    ),
    cmendy = choices_selected(
      choices = variable_choices(ADCM, "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()
  )

```

### 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
aval	<b>[Deprecated]</b> Please use the aval_var argument instead.
aval_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
xaxis	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>c(value, min, max)</code> . 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 <code>c(value, min, max)</code> . 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 <a href="#">shiny::helpText()</a> 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")`.

## 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.

## Examples

```
library(nestcolor)

ADSL <- tmc_ex_adsl
ADVS <- tmc_ex_advs

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADVS = ADVS,
    code = "
      ADSL <- tmc_ex_adsl
      ADVS <- tmc_ex_advs
    "
  ),
  modules = modules(
    tm_g_pp_vitals(
      label = "Vitals",
      dataname = "ADVS",
      parentname = "ADSL",
      patient_col = "USUBJID",
      plot_height = c(600L, 200L, 2000L),
      paramcd = choices_selected(
        choices = variable_choices(ADVS, "PARAMCD"),
        selected = "PARAMCD"
      ),
      xaxis = choices_selected(
        choices = variable_choices(ADVS, "ADY"),
        selected = "ADY"
      ),
      aval_var = choices_selected(
        choices = variable_choices(ADVS, "AVAL"),
        selected = "AVAL"
      )
    )
  )
)
```

```

    )
  )
  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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var, "ADSL"),
    arm_var,
  by_vars,
  grade,
  abnormal = list(low = c("LOW", "LOW LOW"), high = c("HIGH", "HIGH HIGH")),
  id_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    subset = "USUBJID"), selected = "USUBJID", fixed = TRUE),
  baseline_var =
    teal.transform::choices_selected(teal.transform::variable_choices(dataname, subset =
      "BNRIND"), selected = "BNRIND", fixed = TRUE),
  treatment_flag_var =
    teal.transform::choices_selected(teal.transform::variable_choices(dataname, subset =
      "ONTRTFL"), selected = "ONTRTFL", fixed = TRUE),
  treatment_flag = teal.transform::choices_selected("Y"),
  add_total = TRUE,
  total_label = default_total_label(),
  exclude_base_abn = FALSE,
  drop_arm_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
  na_level = default_na_str(),
  basic_table_args = teal.widgets::basic_table_args()
)

```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
by_vars	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names used to split the summary by rows.
grade	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used to specify the abnormality grade. Variable must be factor.
abnormal	(named list) defined by user to indicate what abnormalities are to be displayed.
id_var	( <a href="#">teal.transform::choices_selected()</a> ) object specifying the variable name for subject id.
baseline_var	( <a href="#">teal.transform::choices_selected()</a> ) variable for baseline abnormality grade.
treatment_flag_var	( <a href="#">teal.transform::choices_selected()</a> ) on treatment flag variable.
treatment_flag	( <a href="#">teal.transform::choices_selected()</a> ) value indicating on treatment records in treatment_flag_var.
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 <code>set_default_total_label("new_default")</code> .
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, default 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")`.

### Value

a teal\_module object.

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

app <- init(
  data = data,
  modules = modules(
    tm_t_abnormality(
      label = "Abnormality Table",
      dataname = "ADLB",

```

```

arm_var = choices_selected(
  choices = variable_choices(data[["ADSL"]], subset = c("ARM", "ARMCD")),
  selected = "ARM"
),
add_total = FALSE,
by_vars = choices_selected(
  choices = variable_choices(data[["ADLB"]], subset = c("LBCAT", "PARAM", "AVISIT")),
  selected = c("LBCAT", "PARAM"),
  keep_order = TRUE
),
baseline_var = choices_selected(
  variable_choices(data[["ADLB"]], subset = "BNRIND"),
  selected = "BNRIND", fixed = TRUE
),
grade = choices_selected(
  choices = variable_choices(data[["ADLB"]], subset = "ANRIND"),
  selected = "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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  id_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    subset = "USUBJID"), selected = "USUBJID", fixed = TRUE),
  paramcd,
  atoxgr_var =

```



```

    teal.transform::choices_selected(teal.transform::variable_choices(dataname, subset =
      "ATOXGR"), selected = "ATOXGR", fixed = TRUE),
    worst_high_flag_var =
      teal.transform::choices_selected(teal.transform::variable_choices(dataname, subset =
        "WGRHIFL"), selected = "WGRHIFL", fixed = TRUE),
    worst_low_flag_var =
      teal.transform::choices_selected(teal.transform::variable_choices(dataname, subset =
        "WGRLOFL"), selected = "WGRLOFL", fixed = TRUE),
    worst_flag_indicator = teal.transform::choices_selected("Y"),
    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()
  )

```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
id_var	( <a href="#">teal.transform::choices_selected()</a> ) object specifying the variable name for subject id.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
atoxgr_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as Analysis Toxicity Grade.
worst_high_flag_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as Worst High Grade flag.
worst_low_flag_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as Worst Low Grade flag.

**worst\_flag\_indicator**  
([teal.transform::choices\\_selected\(\)](#))  
value indicating worst grade.

**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")`.

## Value

a `teal_module` object.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
library(dplyr)

ADSL <- tmc_ex_adsl
ADLB <- tmc_ex_adlb %>%
  filter(!AVISIT %in% c("SCREENING", "BASELINE"))

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

```

ADLB = ADLB,
code = "
  ADSL <- tmc_ex_adsl
  ADLB <- tmc_ex_adlb %>%
    filter(!AVISIT %in% c(\"SCREENING\", \"BASELINE\"))
"
),
modules = modules(
  tm_t_abnormality_by_worst_grade(
    label = "Laboratory Test Results with Highest Grade Post-Baseline",
    dataname = "ADLB",
    arm_var = choices_selected(
      choices = variable_choices(ADSL, subset = c("ARM", "ARMCD")),
      selected = "ARM"
    ),
    paramcd = choices_selected(
      choices = value_choices(ADLB, "PARAMCD", "PARAM"),
      selected = c("ALT", "CRP", "IGA")
    ),
    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 AOV01 available [here](#) when multiple endpoints are selected.

## Usage

```

tm_t_ancova(
  label,
  dataname,
  parentname = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "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.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
      TRUE),
    pre_output = NULL,
    post_output = NULL,
    basic_table_args = teal.widgets::basic_table_args()
  )

```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
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 <a href="#">teal.transform::variable_choices()</a> or delayed <a href="#">teal.transform::value_choices()</a> with the elements named ref and comp that the defined the default reference and comparison arms when the arm variable is changed.
aval_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
cov_var	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>interact_var</code> is discrete. If the interaction is not needed, the default option is <code>FALSE</code> .
<code>avisit</code>	<code>(teal.transform::choices_selected())</code> value of analysis visit AVISIT of interest.
<code>paramcd</code>	<code>(teal.transform::choices_selected())</code> object with all available choices and preselected option for the parameter code variable from <code>dataname</code> .
<code>conf_level</code>	<code>(teal.transform::choices_selected())</code> object with all available choices and pre-selected option for the confidence level, each within range of (0, 1).
<code>pre_output</code>	<code>(shiny.tag)</code> optional, with text placed before the output to put the output into context. For example a title.
<code>post_output</code>	<code>(shiny.tag)</code> optional, with text placed after the output to put the output into context. For example the <code>shiny::helpText()</code> elements are useful.
<code>basic_table_args</code>	<code>(basic_table_args)</code> optional object created by <code>teal.widgets::basic_table_args()</code> with settings for the module table. The argument is merged with option <code>teal.basic_table_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-basic-table-arguments", package = "teal.widgets")</code> .

## Details

When a single endpoint is selected, both unadjusted and adjusted comparison are provided. This module 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.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
ADSL <- tmc_ex_adsl
ADQS <- tmc_ex_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 = cdisc_data(
    ADSL = ADSL,
    ADQS = ADQS,
    code = "
      ADSL <- tmc_ex_adsl
      ADQS <- tmc_ex_adqs
    "
  ),
  modules = modules(
    tm_t_ancova(
      label = "ANCOVA Table",
      dataname = "ADQS",
      avisit = choices_selected(
        choices = value_choices(ADQS, "AVISIT"),
        selected = "WEEK 1 DAY 8"
      ),
      arm_var = choices_selected(
        choices = variable_choices(ADSL, c("ARM", "ACTARMCD", "ARMCD")),
        selected = "ARMCD"
      ),
      arm_ref_comp = arm_ref_comp,
      aval_var = choices_selected(
        choices = variable_choices(ADQS, c("CHG", "AVAL")),
        selected = "CHG"
      ),
      cov_var = choices_selected(
        choices = variable_choices(ADQS, c("BASE", "STRATA1", "SEX")),
        selected = "STRATA1"
      ),
      paramcd = choices_selected(
        choices = value_choices(ADQS, "PARAMCD", "PARAM"),
        selected = "FKSI-FWB"
      ),
      interact_var = choices_selected(
        choices = variable_choices(ADQS, c("BASE", "STRATA1", "SEX")),
        selected = "STRATA1"
      )
    )
  )
)

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

```

---

tm_t_binary_outcome	<i>teal Module: Binary Outcome Table</i>
---------------------	--

---

## 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 = ifelse(test = inherits(arm_var, "data_extract_spec"), yes =
    teal.transform::datanames_input(arm_var), no = "ADSL"),
  arm_var,
  arm_ref_comp = NULL,
  paramcd,
  strata_var,
  aval_var = teal.transform::choices_selected(choices =
    teal.transform::variable_choices(dataname, c("AVALC", "SEX")), selected = "AVALC",
    fixed = FALSE),
  conf_level = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
    TRUE),
  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 = default_na_str(),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args()
)
```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
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 <a href="#">teal.transform::variable_choices()</a> or delayed <a href="#">teal.transform::value_choices()</a> with the elements named ref and comp that the defined the default reference and comparison arms when the arm variable is changed.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
strata_var	( <a href="#">teal.transform::choices_selected()</a> ) names of the variables for stratified analysis.
aval_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
conf_level	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the confidence level, each within range of (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.
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: <ul style="list-style-type: none"> <li>• global: a list of settings for overall analysis with 2 named elements method and conf_level.</li> <li>• unstrat: a list of settings for unstratified analysis with 3 named elements method_ci and method_test, and odds. See <a href="#">tern::estimate_proportion_diff()</a>, <a href="#">tern::test_proportion_diff()</a>, and <a href="#">tern::estimate_odds_ratio()</a>, respectively, for options and details on how these settings are implemented in the analysis.</li> <li>• strat: a list of settings for stratified analysis with elements method_ci and method_test. See <a href="#">tern::estimate_proportion_diff()</a> and <a href="#">tern::test_proportion_diff()</a>, respectively, for options and details on how these settings are implemented in the analysis.</li> </ul>



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 <code>set_default_total_label("new_default")</code> .
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 <code>set_default_na_str("new_default")</code> .
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 <code>shiny::helpText()</code> elements are useful.
basic_table_args	(basic_table_args) optional object created by <code>teal.widgets::basic_table_args()</code> with settings for the module table. The argument is merged with option <code>teal.basic_table_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-basic-table-arguments", package = "teal.widgets")</code> .

## 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.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
library(dplyr)

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")

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 = cdisc_data(
    ADSL = ADSL,
    ADRS = ADRS,
    code = "
      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")
    "
  ),
  modules = modules(
    tm_t_binary_outcome(
      label = "Responders",
      dataname = "ADRS",
      paramcd = choices_selected(
        choices = value_choices(ADRS, "PARAMCD", "PARAM"),
        selected = "BESRSPI"
      ),
    ),
    arm_var = choices_selected(
      choices = variable_choices(ADRS, c("ARM", "ARMCD", "ACTARMCD")),
      selected = "ARM"
    ),
    arm_ref_comp = arm_ref_comp,
    strata_var = choices_selected(
      choices = variable_choices(ADRS, 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)")
  )
)
)
)
)
)
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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  arm_ref_comp = NULL,
  paramcd,
  cov_var,
  strata_var,
  aval_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AVAL"), "AVAL", fixed = TRUE),
  cnsr_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "CNSR"), "CNSR", fixed = TRUE),
  multivariate = TRUE,
  na_level = default_na_str(),
  conf_level = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
    TRUE),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args()
)

```

**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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
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 <a href="#">teal.transform::variable_choices()</a> or delayed <a href="#">teal.transform::value_choices()</a> with the elements named ref and comp that the defined the default reference and comparison arms when the arm variable is changed.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
cov_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the covariates variables.
strata_var	( <a href="#">teal.transform::choices_selected()</a> ) names of the variables for stratified analysis.
aval_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
cnsr_var	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>set_default_na_str("new_default")</code> .
conf_level	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the confidence level, each within range of (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")`.

## 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 module 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.

## 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
})
datanames <- c("ADSL", "ADTTE")
datanames(data) <- datanames
join_keys(data) <- default_cdisc_join_keys[datanames]

app <- init(
  data = data,
  modules = modules(
    tm_t_coxreg(
      label = "Cox Reg.",
      dataname = "ADTTE",
      arm_var = choices_selected(c("ARM", "ARMCD", "ACTARMCD"), "ARM"),
      arm_ref_comp = arm_ref_comp,
      paramcd = choices_selected(
        value_choices(data[["ADTTE"]], "PARAMCD", "PARAM"), "OS"
      ),
      strata_var = choices_selected(
        c("COUNTRY", "STRATA1", "STRATA2"), "STRATA1"
      ),
      cov_var = choices_selected(
        c("AGE", "BMRKR1", "BMRKR2", "REGION1"), "AGE"
      ),
      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.

## Dataset fabrication
## -----

data <- teal_data()

```

```

data <- within(data, {
  library(dplyr)
  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")
  )
})

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

## `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 = choices_selected(c("ARMCD"), "ARMCD"),
      arm_ref_comp = arm_ref_comp,
      paramcd = choices_selected(
        value_choices(data[["ADTTE"]], "PARAMCD", "PARAM"), "OS"
      ),
      strata_var = choices_selected(c("INST"), NULL),

```

```

      cov_var = choices_selected(c("SEX", "AGE"), "SEX"),
      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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  hlt,
  llt,
  add_total = TRUE,
  total_label = default_total_label(),
  na_level = 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()
)

```

### 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	( <code>teal.transform::choices_selected()</code> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
hlt	( <code>teal.transform::choices_selected()</code> ) name of the variable with high level term for events.
llt	( <code>teal.transform::choices_selected()</code> ) 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 <code>set_default_total_label("new_default")</code> .
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 <code>set_default_na_str("new_default")</code> .
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")`.

## Value

a `teal_module` object.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
ADSL <- tmc_ex_adsl
ADAE <- tmc_ex_adae

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADAE = ADAE,
    code = "
      ADSL <- tmc_ex_adsl
      ADAE <- tmc_ex_adae
    "
  ),
  modules = modules(
    tm_t_events(
      label = "Adverse Event Table",
      dataname = "ADAE",
      arm_var = choices_selected(c("ARM", "ARMCD"), "ARM"),
      llt = choices_selected(
        choices = variable_choices(ADAE, c("AETERM", "AEDECOD")),
        selected = c("AEDECOD")
      ),
      hlt = choices_selected(
        choices = variable_choices(ADAE, 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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var, "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 = default_na_str(),
    drop_arm_levels = TRUE,
    pre_output = NULL,
    post_output = NULL,
    basic_table_args = teal.widgets::basic_table_args()
)

```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
hlt	( <a href="#">teal.transform::choices_selected()</a> ) name of the variable with high level term for events.
llt	( <a href="#">teal.transform::choices_selected()</a> ) name of the variable with low level term for events.
grade	(character) name of the severity level variable.
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 <code>set_default_total_label("new_default")</code> .
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 <code>set_default_na_str("new_default")</code> .
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 <a href="#">shiny::helpText()</a> 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").
```

## Value

a teal\_module object.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
data <- teal_data()
data <- within(data, {
  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
})

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

app <- init(
  data = data,
  modules = modules(
    tm_t_events_by_grade(
      label = "Adverse Events by Grade Table",
      dataname = "ADAE",
      arm_var = choices_selected(c("ARM", "ARMCD"), "ARM"),
      llt = choices_selected(
        choices = variable_choices(data[["ADAE"]], c("AETERM", "AEDECOD")),
        selected = c("AEDECOD")
      ),
      hlt = choices_selected(
        choices = variable_choices(data[["ADAE"]], c("AEBODSYS", "AESOC")),
        selected = "AEBODSYS"
      ),
      grade = choices_selected(
        choices = variable_choices(data[["ADAE"]], c("AETOXGR", "AESEV")),
        selected = "AETOXGR"
      )
    )
  )
)
```

```

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

```

---

tm_t_events_patyear	<i>teal Module: Event Rates Adjusted for Patient-Years</i>
---------------------	--

---

## Description

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

## Usage

```

tm_t_events_patyear(
  label,
  dataname,
  parentname = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var, "ADSL"),
    arm_var,
    events_var,
    paramcd,
    aval_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
      "AVAL"), "AVAL", fixed = TRUE),
    avalu_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
      "AVALU"), "AVALU", fixed = TRUE),
    add_total = TRUE,
    total_label = default_total_label(),
    na_level = default_na_str(),
    conf_level = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
      TRUE),
    drop_arm_levels = TRUE,
    pre_output = NULL,
    post_output = NULL,
    basic_table_args = teal.widgets::basic_table_args()
)

```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
events_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the variable with all event counts.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
aval_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
avalu_var	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>set_default_total_label("new_default")</code> .
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 <code>set_default_na_str("new_default")</code> .
conf_level	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the confidence level, each within range of (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 <a href="#">shiny::helpText()</a> 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")`.

## Value

a teal\_module object.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
library(dplyr)
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))

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADAETTE = ADAETTE,
    code = "
      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))
    "
  ),
  modules = modules(
    tm_t_events_patyear(
      label = "AE Rate Adjusted for Patient-Years At Risk Table",
      dataname = "ADAETTE",
      arm_var = choices_selected(
        choices = variable_choices(ADSL, c("ARM", "ARMCD")),
        selected = "ARMCD"
      ),
      add_total = TRUE,
      events_var = choices_selected(
        choices = variable_choices(ADAETTE, "n_events"),
        selected = "n_events",
        fixed = TRUE
      ),
      paramcd = choices_selected(
```



```

        choices = value_choices(ADAETTE, "PARAMCD", "PARAM"),
        selected = "AETTE1"
      )
    )
  )
}
if (interactive()) {
  shinyApp(app$ui, app$server)
}

```

---

tm_t_events_summary	<i>teal Module: Adverse Events Summary</i>
---------------------	--

---

## Description

This module produces an adverse events summary table.

## Usage

```

tm_t_events_summary(
  label,
  dataname,
  parentname = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  flag_var_anl = NULL,
  flag_var_aesl = NULL,
  dthfl_var =
    teal.transform::choices_selected(teal.transform::variable_choices(parentname,
      "DTHFL"), "DTHFL", fixed = TRUE),
  dcsreas_var =
    teal.transform::choices_selected(teal.transform::variable_choices(parentname,
      "DCSREAS"), "DCSREAS", fixed = TRUE),
  llt = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AEDECOD"), "AEDECOD", fixed = TRUE),
  aeseq_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AESEQ"), "AESEQ", fixed = TRUE),
  add_total = TRUE,
  total_label = default_total_label(),
  na_level = default_na_str(),
  count_subj = TRUE,
  count_pt = TRUE,
  count_events = TRUE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args()
)

```

**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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
flag_var_anl	( <a href="#">teal.transform::choices_selected()</a> 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_aesl	( <a href="#">teal.transform::choices_selected()</a> 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	( <a href="#">teal.transform::choices_selected()</a> ) 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	( <a href="#">teal.transform::choices_selected()</a> ) object 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	( <a href="#">teal.transform::choices_selected()</a> ) name of the variable with low level term for events.
aeseq_var	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>set_default_total_label("new_default")</code> .
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 <code>set_default_na_str("new_default")</code> .

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 ll1t). 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 <a href="#">shiny::helpText()</a> elements are useful.
basic_table_args	(basic_table_args) optional object created by <a href="#">teal.widgets::basic_table_args()</a> with settings for the module table. The argument is merged with option <code>teal.basic_table_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-basic-table-arguments", package = "teal.widgets")</code> .

## Value

a `teal_module` object.

## 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 %>%
    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/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_")]
})

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

app <- init(
  data = data,
  modules = modules(
    tm_t_events_summary(
      label = "Adverse Events Summary",
      dataname = "ADAE",
      arm_var = choices_selected(
        choices = variable_choices("ADSL", c("ARM", "ARMCD")),
        selected = "ARM"
      ),
    ),
    flag_var_anl = choices_selected(
      choices = variable_choices("ADAE", data[["ae_anl_vars"]]),
      selected = data[["ae_anl_vars"]][1],
      keep_order = TRUE,
      fixed = FALSE
    ),
    flag_var_aesi = choices_selected(
      choices = variable_choices("ADAE", data[["aesi_vars"]]),
      selected = data[["aesi_vars"]][1],
      keep_order = 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 = ifelse(inherits(col_by_var, "data_extract_spec"),
    teal.transform::datanames_input(col_by_var), "ADSL"),
  row_by_var,
  col_by_var,
  paramcd = teal.transform::choices_selected(choices =
    teal.transform::value_choices(dataname, "PARAMCD", "PARAM"), selected = "TDURD"),
  paramcd_label = "PARAM",
  id_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    subset = "USUBJID"), selected = "USUBJID", fixed = TRUE),
  parcat,
  aval_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    subset = "AVAL"), selected = "AVAL", fixed = TRUE),
  avalu_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    subset = "AVALU"), selected = "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 = default_na_str(),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args()
)

```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used to split rows.
col_by_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used to split columns.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
paramcd_label	(character) the column from the dataset where the value will be used to label the argument paramcd.
id_var	( <a href="#">teal.transform::choices_selected()</a> ) object specifying the variable name for subject id.
parcat	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for parameter category values.
aval_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
avalu_var	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>set_default_total_label("new_default")</code> .
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 <code>set_default_na_str("new_default")</code> .

`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")`.

### Value

a `teal_module` object.

### 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
})

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

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

```

modules = modules(
  tm_t_exposure(
    label = "Duration of Exposure Table",
    dataname = "ADEX",
    paramcd = choices_selected(
      choices = value_choices(data[["ADEX"]], "PARAMCD", "PARAM"),
      selected = "TDURD"
    ),
    col_by_var = choices_selected(
      choices = variable_choices(data[["ADEX"]], subset = c("SEX", "ARM")),
      selected = "SEX"
    ),
    row_by_var = choices_selected(
      choices = variable_choices(data[["ADEX"]], subset = c("RACE", "REGION1", "STRATA1", "SEX")),
      selected = "RACE"
    ),
    parcat = choices_selected(
      choices = value_choices(data[["ADEX"]], "PARCAT2"),
      selected = "Drug A"
    ),
    add_total = FALSE
  )
),
filter = teal_slices(teal_slice("ADSL", "SAFFL", selected = "Y"))
)
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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var = NULL,
  arm_ref_comp = NULL,
  paramcd,
  cov_var = NULL,

```



```

    avalc_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
      "AVALC"), "AVALC", fixed = TRUE),
    conf_level = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order =
      TRUE),
    pre_output = NULL,
    post_output = NULL,
    basic_table_args = teal.widgets::basic_table_args()
  )

```

## 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	( <a href="#">teal.transform::choices_selected()</a> 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 <a href="#">teal.transform::variable_choices()</a> or delayed <a href="#">teal.transform::value_choices()</a> with the elements named ref and comp that the defined the default reference and comparison arms when the arm variable is changed.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
cov_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the covariates variables.
avalc_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the analysis variable (categorical).
conf_level	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the confidence level, each within range of (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 <a href="#">shiny::helpText()</a> 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")`.

## Value

a `teal_module` object.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
library(dplyr)

ADSL <- tmc_ex_adsl
ADRS <- tmc_ex_adrs %>%
  filter(PARAMCD %in% c("BESRSPI", "INVET"))

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 = cdisc_data(
    ADSL = ADSL,
    ADRS = ADRS,
    code = "
      ADSL <- tmc_ex_adsl
      ADRS <- tmc_ex_adrs %>%
        filter(PARAMCD %in% c(\"BESRSPI\", \"INVET\"))
    "
  ),
  modules = modules(
    tm_t_logistic(
      label = "Logistic Regression",
      dataname = "ADRS",
      arm_var = choices_selected(
        choices = variable_choices(ADRS, c("ARM", "ARMCD")),
        selected = "ARM"
      )
    )
  )
)
```

```

    ),
    arm_ref_comp = arm_ref_comp,
    paramcd = choices_selected(
      choices = value_choices(ADRS, "PARAMCD", "PARAM"),
      selected = "BESRSPI"
    ),
    cov_var = choices_selected(
      choices = c("SEX", "AGE", "BMRKR1", "BMRKR2"),
      selected = "SEX"
    )
  )
)
)
)
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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  seq_var,
  hlt,
  llt,
  add_total = TRUE,
  total_label = default_total_label(),
  na_level = default_na_str(),
  event_type = "event",
  drop_arm_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args()
)

```

**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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
seq_var	( <a href="#">teal.transform::choices_selected()</a> ) 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	( <a href="#">teal.transform::choices_selected()</a> ) name of the variable with high level term for events.
llt	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>set_default_total_label("new_default")</code> .
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 <code>set_default_na_str("new_default")</code> .
event_type	(character) type of event that is summarized (e.g. adverse event, treatment). Default is "event".
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")`.

## Value

a teal\_module object.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
ADSL <- tmc_ex_adsl
ADCM <- tmc_ex_adcm
adcm_keys <- c("STUDYID", "USUBJID", "ASTDTM", "CMSEQ", "ATC1", "ATC2", "ATC3", "ATC4")

join_keys <- default_cdisc_join_keys[c("ADSL", "ADCM")]
join_keys["ADCM", "ADCM"] <- adcm_keys

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADCM = ADCM,
    code = "
      ADSL <- tmc_ex_adsl
      ADCM <- tmc_ex_adcm
    ",
    join_keys = join_keys
  ),
  modules = modules(
    tm_t_mult_events(
      label = "Concomitant Medications by Medication Class and Preferred Name",
      dataname = "ADCM",
      arm_var = choices_selected(c("ARM", "ARMCD"), "ARM"),
      seq_var = choices_selected("CMSEQ", selected = "CMSEQ", fixed = TRUE),
      hlt = choices_selected(
        choices = variable_choices(ADCM, c("ATC1", "ATC2", "ATC3", "ATC4")),
        selected = c("ATC1", "ATC2", "ATC3", "ATC4")
      ),
    ),
    llt = choices_selected(
      choices = variable_choices(ADCM, c("CMDECOD")),
      selected = c("CMDECOD")
    ),
  ),
)
```

```

      add_total = TRUE,
      event_type = "treatment"
    )
  )
}
if (interactive()) {
  shinyApp(app$ui, app$server)
}

```

---

tm_t_pp_basic_info	<i>teal Module: Patient Profile Basic Info</i>
--------------------	--

---

## 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
)

```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for 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 <a href="#">shiny::helpText()</a> elements are useful.

**Value**

a teal\_module object.

**Examples**

```
ADSL <- tmc_ex_adsl

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    code = "ADSL <- tmc_ex_adsl"
  ),
  modules = modules(
    tm_t_pp_basic_info(
      label = "Basic Info",
      dataname = "ADSL",
      patient_col = "USUBJID",
      vars = choices_selected(
        choices = variable_choices(ADSL),
        selected = c("ARM", "AGE", "SEX", "COUNTRY", "RACE", "EOSSTT")
      )
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}
```

---

tm_t_pp_laboratory	<i>teal Module: Patient Profile Laboratory Table</i>
--------------------	--

---

**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",
  timepoints = 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
  )

```

## 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.
timepoints	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the time variable from dataname.
aval	<b>[Deprecated]</b> Please use the <code>aval_var</code> argument instead.
aval_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
avalu	<b>[Deprecated]</b> Please use the <code>avalu_var</code> argument instead.
avalu_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the analysis unit variable.
param	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the PARAM variable from dataname.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
anrind	( <a href="#">teal.transform::choices_selected()</a> ) 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 <a href="#">shiny::helpText()</a> elements are useful.

## Value

a teal\_module object.



**Examples**

```

ADSL <- tmc_ex_adsl
ADLB <- tmc_ex_adlb

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADLB = ADLB,
    code = "
      ADSL <- tmc_ex_adsl
      ADLB <- tmc_ex_adlb
    "
  ),
  modules = modules(
    tm_t_pp_laboratory(
      label = "Vitals",
      dataname = "ADLB",
      patient_col = "USUBJID",
      paramcd = choices_selected(
        choices = variable_choices(ADLB, "PARAMCD"),
        selected = "PARAMCD"
      ),
      param = choices_selected(
        choices = variable_choices(ADLB, "PARAM"),
        selected = "PARAM"
      ),
      timepoints = choices_selected(
        choices = variable_choices(ADLB, "ADY"),
        selected = "ADY"
      ),
      anrind = choices_selected(
        choices = variable_choices(ADLB, "ANRIND"),
        selected = "ANRIND"
      ),
      aval_var = choices_selected(
        choices = variable_choices(ADLB, "AVAL"),
        selected = "AVAL"
      ),
      avalu_var = choices_selected(
        choices = variable_choices(ADLB, "AVALU"),
        selected = "AVALU"
      )
    )
  )
)
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
)
```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the MHTERM variable from dataname.
mhbodsys	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the MHBODSYS variable from dataname.
mhdistat	( <a href="#">teal.transform::choices_selected()</a> ) 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.

## Value

a teal\_module object.

## Examples

```
ADSL <- tmc_ex_adsl
ADMH <- tmc_ex_admh

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADMH = ADMH,
    code = "
      ADSL <- tmc_ex_adsl
      ADMH <- tmc_ex_admh
    "
  ),
  modules = modules(
    tm_t_pp_medical_history(
      label = "Medical History",
      dataname = "ADMH",
      parentname = "ADSL",
      patient_col = "USUBJID",
      mhterm = choices_selected(
        choices = variable_choices(ADMH, c("MHTERM")),
        selected = "MHTERM"
      ),
      mhbodsys = choices_selected(
        choices = variable_choices(ADMH, "MHBODSYS"),
        selected = "MHBODSYS"
      ),
      mhdistat = choices_selected(
        choices = variable_choices(ADMH, "MHDISTAT"),
        selected = "MHDISTAT"
      )
    )
  )
)
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
)
```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the ATIREL variable from dataname.
cmdecod	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMDECOD variable from dataname.
cmindc	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the CMINDC variable from dataname.
cmstdy	( <a href="#">teal.transform::choices_selected()</a> ) 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.

## Value

a teal\_module object.

## Examples

```
library(dplyr)

ADCM <- tmc_ex_adcm
ADSL <- tmc_ex_adsl %>% filter(USUBJID %in% ADCM$USUBJID)
ADCM$CMASTDTM <- ADCM$ASTDTM
ADCM$CMAENDTM <- ADCM$AENDTM
adcm_keys <- c("STUDYID", "USUBJID", "ASTDTM", "CMSEQ", "ATC1", "ATC2", "ATC3", "ATC4")

join_keys <- default_cdisc_join_keys[c("ADSL", "ADCM")]
join_keys["ADCM", "ADCM"] <- adcm_keys

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADCM = ADCM,
    code = "
      ADCM <- tmc_ex_adcm
      ADSL <- tmc_ex_adsl %>% filter(USUBJID %in% ADCM$USUBJID)
      ADCM$CMASTDTM <- ADCM$ASTDTM
      ADCM$CMAENDTM <- ADCM$AENDTM
    ",
    join_keys = join_keys
  ),
  modules = modules(
    tm_t_pp_prior_medication(
      label = "Prior Medication",
      dataname = "ADCM",
      parentname = "ADSL",
      patient_col = "USUBJID",
      atirel = choices_selected(
        choices = variable_choices(ADCM, "ATIREL"),
        selected = "ATIREL"
      ),
      cmdecod = choices_selected(
        choices = variable_choices(ADCM, "CMDECOD"),
        selected = "CMDECOD"
      ),
      cmindc = choices_selected(
        choices = variable_choices(ADCM, "CMINDC"),
```

```

        selected = "CMINDC"
      ),
      cmstdy = choices_selected(
        choices = variable_choices(ADCM, "ASTDY"),
        selected = "ASTDY"
      )
    )
  )
}
if (interactive()) {
  shinyApp(app$ui, app$server)
}

```

---

tm_t_shift_by_arm	<i>teal Module: Shift by Arm</i>
-------------------	----------------------------------

---

## Description

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

## Usage

```

tm_t_shift_by_arm(
  label,
  dataname,
  parentname = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  paramcd,
  visit_var,
  aval_var,
  base_var = lifecycle::deprecated(),
  baseline_var,
  treatment_flag_var =
    teal.transform::choices_selected(teal.transform::variable_choices(dataname, subset =
      "ONTRTFL"), selected = "ONTRTFL"),
  treatment_flag = teal.transform::choices_selected("Y"),
  useNA = c("ifany", "no"),
  na_level = 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()
)

```

**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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
visit_var	( <a href="#">teal.transform::choices_selected()</a> ) 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
base_var	<b>[Deprecated]</b> Please use the baseline_var argument instead.
baseline_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable values that can be used as baseline_var.
treatment_flag_var	( <a href="#">teal.transform::choices_selected()</a> ) on treatment flag variable.
treatment_flag	( <a href="#">teal.transform::choices_selected()</a> ) value indicating on treatment records in treatment_flag_var.
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 <code>set_default_na_str("new_default")</code> .
add_total	(logical) whether to include row 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 <code>set_default_total_label("new_default")</code> .
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")`.

## Value

a teal\_module object.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
ADSL <- tmc_ex_adsl
ADEG <- tmc_ex_adeg

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADEG = ADEG,
    code = "
      ADSL <- tmc_ex_adsl
      ADEG <- tmc_ex_adeg
    "
  ),
  modules = modules(
    tm_t_shift_by_arm(
      label = "Shift by Arm Table",
      dataname = "ADEG",
      arm_var = choices_selected(
        variable_choices(ADSL, subset = c("ARM", "ARMCD")),
        selected = "ARM"
      ),
      paramcd = choices_selected(
        value_choices(ADEG, "PARAMCD"),
        selected = "HR"
      ),
      visit_var = choices_selected(
        value_choices(ADEG, "AVISIT"),
        selected = "POST-BASELINE MINIMUM"
      ),
      aval_var = choices_selected(
        variable_choices(ADEG, subset = "ANRIND"),
        selected = "ANRIND", fixed = TRUE
      )
    )
  )
)
```



```

    ),
    baseline_var = choices_selected(
      variable_choices(ADEG, subset = "BNRIND"),
      selected = "BNRIND", fixed = TRUE
    ),
    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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  paramcd,
  aval_var,
  base_var = lifecycle::deprecated(),
  baseline_var,
  worst_flag_var,
  worst_flag,
  treatment_flag_var = teal.transform::choices_selected(choices =
    teal.transform::variable_choices(dataname, subset = "ONTRTFL"), selected = "ONTRTFL"),
  treatment_flag = teal.transform::choices_selected("Y"),
  useNA = c("ifany", "no"),
  na_level = 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()
)

```

**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	( <code>teal.transform::choices_selected()</code> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
paramcd	( <code>teal.transform::choices_selected()</code> ) object with all available choices and preselected option for the parameter code variable from dataname.
aval_var	( <code>teal.transform::choices_selected()</code> ) object with all available choices and pre-selected option for the analysis variable.
base_var	<b>[Deprecated]</b> Please use the baseline_var argument instead.
baseline_var	( <code>teal.transform::choices_selected()</code> ) object with all available choices and preselected option for variable values that can be used as baseline_var.
worst_flag_var	( <code>teal.transform::choices_selected()</code> ) 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	( <code>teal.transform::choices_selected()</code> ) on treatment flag variable.
treatment_flag	( <code>teal.transform::choices_selected()</code> ) value indicating on treatment records in treatment_flag_var.
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 <code>set_default_na_str("new_default")</code> .
add_total	(logical) whether to include row 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 <code>set_default_total_label("new_default")</code> .

`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")`.

## Value

a teal\_module object.

## Examples

```
ADSL <- tmc_ex_adsl
ADEG <- tmc_ex_adeg

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADEG = ADEG,
    code = "
      ADSL <- tmc_ex_adsl
      ADEG <- tmc_ex_adeg
    "
  ),
  modules = modules(
    tm_t_shift_by_arm_by_worst(
      label = "Shift by Arm Table",
      dataname = "ADEG",
      arm_var = choices_selected(
        variable_choices(ADSL, subset = c("ARM", "ARMCD")),
        selected = "ARM"
      ),
      paramcd = choices_selected(
        value_choices(ADEG, "PARAMCD"),
        selected = "ECGINTP"
      ),
      worst_flag_var = choices_selected(
        variable_choices(ADEG, c("WORS02FL", "WORS01FL")),
        selected = "WORS02FL"
      ),
      worst_flag = choices_selected(
        value_choices(ADEG, "WORS02FL"),
        selected = "Y", fixed = TRUE
      )
    )
  )
)
```

```

    ),
    aval_var = choices_selected(
      variable_choices(ADEG, c("AVALC", "ANRIND")),
      selected = "AVALC"
    ),
    baseline_var = choices_selected(
      variable_choices(ADEG, c("BASEC", "BNRIND")),
      selected = "BASEC"
    ),
    useNA = "ifany"
  )
)
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}

```

---

tm_t_shift_by_grade	<i>teal Module: Grade Summary Table</i>
---------------------	---

---

## 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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  visit_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    subset = "AVISIT"), selected = "AVISIT", fixed = TRUE),
  paramcd,
  worst_flag_var =
    teal.transform::choices_selected(teal.transform::variable_choices(dataname, subset =
      c("WGRLOVFL", "WGRLOFL", "WGRHIVFL", "WGRHIFL")), selected = "WGRLOVFL"),
  worst_flag_indicator =
    teal.transform::choices_selected(teal.transform::value_choices(dataname, "WGRLOVFL"),
      selected = "Y", fixed = TRUE),
  anl_toxgrade_var =
    teal.transform::choices_selected(teal.transform::variable_choices(dataname, subset =
      c("ATOXGR")), selected = c("ATOXGR"), fixed = TRUE),
  base_toxgrade_var =
    teal.transform::choices_selected(teal.transform::variable_choices(dataname, subset =
      c("BTOXGR")), selected = c("BTOXGR"), fixed = TRUE),

```

```

id_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
  subset = "USUBJID"), selected = "USUBJID", fixed = TRUE),
add_total = FALSE,
total_label = default_total_label(),
drop_arm_levels = TRUE,
pre_output = NULL,
post_output = NULL,
na_level = default_na_str(),
code_missing_baseline = FALSE,
basic_table_args = teal.widgets::basic_table_args()
)

```

## 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.transform::choices_selected()) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
visit_var	(teal.transform::choices_selected()) 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.transform::choices_selected()) object with all available choices and preselected option for the parameter code variable from dataname.
worst_flag_var	(teal.transform::choices_selected()) object with all available choices and preselected option for variable names that can be used as worst flag variable.
worst_flag_indicator	(teal.transform::choices_selected()) value indicating worst grade.
anl_toxgrade_var	(teal.transform::choices_selected()) variable for analysis toxicity grade.
base_toxgrade_var	(teal.transform::choices_selected()) variable for baseline toxicity grade.
id_var	(teal.transform::choices_selected()) 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 <code>set_default_total_label("new_default")</code> .
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 <code>shiny::helpText()</code> 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 <code>set_default_na_str("new_default")</code> .
code_missing_baseline	(logical) whether missing baseline grades should be counted as grade 0.
basic_table_args	(basic_table_args) optional object created by <code>teal.widgets::basic_table_args()</code> with settings for the module table. The argument is merged with option <code>teal.basic_table_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-basic-table-arguments", package = "teal.widgets")</code> .

**Value**

a teal\_module object.

**See Also**

The [TLG Catalog](#) where additional example apps implementing this module can be found.

**Examples**

```
ADSL <- tmc_ex_adsl
ADLB <- tmc_ex_adlb

app <- init(
```

```

data = cdisc_data(
  ADSL = ADSL,
  ADLB = ADLB,
  code = "
    ADSL <- tmc_ex_adsl
    ADLB <- tmc_ex_adlb
  "
),
modules = modules(
  tm_t_shift_by_grade(
    label = "Grade Laboratory Abnormality Table",
    dataname = "ADLB",
    arm_var = choices_selected(
      choices = variable_choices(ADSL, subset = c("ARM", "ARMCD")),
      selected = "ARM"
    ),
    paramcd = choices_selected(
      choices = value_choices(ADLB, "PARAMCD", "PARAM"),
      selected = "ALT"
    ),
    ),
    worst_flag_var = choices_selected(
      choices = variable_choices(ADLB, subset = c("WGRLOVFL", "WGRLOFL", "WGRHIVFL", "WGRHIFL")),
      selected = c("WGRLOVFL")
    ),
    ),
    worst_flag_indicator = choices_selected(
      value_choices(ADLB, "WGRLOVFL"),
      selected = "Y", fixed = TRUE
    ),
    ),
    anl_toxgrade_var = choices_selected(
      choices = variable_choices(ADLB, subset = c("ATOXGR")),
      selected = c("ATOXGR"),
      fixed = TRUE
    ),
    ),
    base_toxgrade_var = choices_selected(
      choices = variable_choices(ADLB, subset = c("BTOXGR")),
      selected = c("BTOXGR"),
      fixed = TRUE
    ),
    ),
    add_total = FALSE
  )
),
filter = teal_slices(teal_slice("ADSL", "SAFFL", selected = "Y"))
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}

```

## Description

This module produces an adverse events table by Standardized MedDRA Query.

## Usage

```
tm_t_smq(
  label,
  dataname,
  parentname = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  id_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    subset = "USUBJID"), selected = "USUBJID", fixed = TRUE),
  llt,
  add_total = TRUE,
  total_label = default_total_label(),
  sort_criteria = c("freq_desc", "alpha"),
  drop_arm_levels = TRUE,
  na_level = default_na_str(),
  smq_varlabel = "Standardized MedDRA Query",
  baskets,
  scopes,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args()
)
```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
id_var	( <a href="#">teal.transform::choices_selected()</a> ) object specifying the variable name for subject id.
llt	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>add_total</code> ). Defaults to "All Patients". To set a new default total_label to apply in all modules, run <code>set_default_total_label("new_default")</code> .
sort_criteria	(character) how to sort the final table. Default option <code>freq_desc</code> sorts on column <code>sort_freq_col</code> by decreasing number of patients with event. Alternative option <code>alpha</code> sorts events alphabetically.
drop_arm_levels	(logical) whether to drop unused levels of <code>arm_var</code> . If TRUE, <code>arm_var</code> levels are set to those used in the <code>dataname</code> dataset. If FALSE, <code>arm_var</code> levels are set to those used in the <code>parentname</code> dataset. If <code>dataname</code> and <code>parentname</code> are the same, then <code>drop_arm_levels</code> 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 <code>set_default_na_str("new_default")</code> .
smq_varlabel	(character) label to use for new column SMQ created by <code>tern::h_stack_by_baskets()</code> .
baskets	( <code>teal.transform::choices_selected()</code> ) object with all available choices and preselected options for standardized/customized queries.
scopes	( <code>teal.transform::choices_selected()</code> ) object with all available choices for the scopes of standardized queries.
pre_output	( <code>shiny.tag</code> ) optional, with text placed before the output to put the output into context. For example a title.
post_output	( <code>shiny.tag</code> ) optional, with text placed after the output to put the output into context. For example the <code>shiny::helpText()</code> elements are useful.
basic_table_args	( <code>basic_table_args</code> ) optional object created by <code>teal.widgets::basic_table_args()</code> with settings for the module table. The argument is merged with option <code>teal.basic_table_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-basic-table-arguments", package = "teal.widgets")</code> .

## Value

a `teal_module` object.

## 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

  names_baskets <- grep("^(SMQ|CQ).*NAM$", names(ADAE), value = TRUE)
  names_scopes <- grep("^SMQ.*SC$", names(ADAE), value = TRUE)

  cs_baskets <- choices_selected(
    choices = variable_choices(ADAE, subset = names_baskets),
    selected = names_baskets
  )

  cs_scopes <- choices_selected(
    choices = variable_choices(ADAE, subset = names_scopes),
    selected = names_scopes,
    fixed = TRUE
  )
})

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

app <- init(
  data = data,
  modules = modules(
    tm_t_smq(
      label = "Adverse Events by SMQ Table",
      dataname = "ADAE",
      arm_var = choices_selected(
        choices = variable_choices(data[["ADSL"]], subset = c("ARM", "SEX")),
        selected = "ARM"
      ),
      add_total = FALSE,
      baskets = data[["cs_baskets"]],
      scopes = data[["cs_scopes"]],
      llt = choices_selected(
        choices = variable_choices(data[["ADAE"]], subset = c("AEDECOD")),
        selected = "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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  summarize_vars,
  add_total = TRUE,
  total_label = default_total_label(),
  useNA = c("ifany", "no"),
  na_level = default_na_str(),
  numeric_stats = c("n", "mean_sd", "mean_ci", "median", "median_ci", "quantiles",
    "range", "geom_mean"),
  denominator = c("N", "n", "omit"),
  drop_arm_levels = TRUE,
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args()
)
```

**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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
summarize_vars	( <a href="#">teal.transform::choices_selected()</a> ) 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 <code>add_total</code> ). Defaults to "All Patients". To set a new default total_label to apply in all modules, run <code>set_default_total_label("new_default")</code> .
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 <code>set_default_na_str("new_default")</code> .
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.
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 <code>shiny::helpText()</code> elements are useful.
basic_table_args	(basic_table_args) optional object created by <code>teal.widgets::basic_table_args()</code> with settings for the module table. The argument is merged with option <code>teal.basic_table_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-basic-table-arguments", package = "teal.widgets")</code> .

### Value

a teal\_module object.

### 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.
ADSL <- tmc_ex_adsl
ADSL$EOSDY[1] <- NA_integer_

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    code = "
      ADSL <- tmc_ex_adsl
      ADSL$EOSDY[1] <- NA_integer_
    "
  ),
  modules = modules(
    tm_t_summary(
      label = "Demographic Table",
      dataname = "ADSL",
      arm_var = choices_selected(c("ARM", "ARMCD"), "ARM"),
      add_total = TRUE,
      summarize_vars = choices_selected(
        c("SEX", "RACE", "BMRKR2", "EOSDY", "DCSREAS", "AGE"),
        c("SEX", "RACE")
      ),
      useNA = "ifany"
    )
  )
)
if (interactive()) {
  shinyApp(app$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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  by_vars,
  summarize_vars,
```

```

id_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
  subset = "USUBJID"), selected = "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 = default_na_str(),
numeric_stats = c("n", "mean_sd", "median", "range"),
denominator = teal.transform::choices_selected(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()
)

```

## 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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
by_vars	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names used to split the summary by rows.
summarize_vars	( <a href="#">teal.transform::choices_selected()</a> ) names of the variables that should be summarized.
id_var	( <a href="#">teal.transform::choices_selected()</a> ) object specifying the variable name for subject id.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
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 <code>set_default_total_label("new_default")</code> .
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 <code>set_default_na_str("new_default")</code> .
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.
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 <code>shiny::helpText()</code> elements are useful.
basic_table_args	(basic_table_args) optional object created by <code>teal.widgets::basic_table_args()</code> with settings for the module table. The argument is merged with option <code>teal.basic_table_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-basic-table-arguments", package = "teal.widgets")</code> .

**Value**

a teal\_module object.

**See Also**

The [TLG Catalog](#) where additional example apps implementing this module can be found.

**Examples**

```
ADSL <- tmc_ex_adsl
ADLB <- tmc_ex_adlb

app <- init(
  data = cdisc_data(
    ADSL = ADSL,
    ADLB = ADLB,
    code = "
      ADSL <- tmc_ex_adsl
      ADLB <- tmc_ex_adlb
    "
  ),
  modules = modules(
    tm_t_summary_by(
      label = "Summary by Row Groups Table",
      dataname = "ADLB",
      arm_var = choices_selected(
        choices = variable_choices(ADSL, c("ARM", "ARMCD")),
        selected = "ARM"
      ),
      add_total = TRUE,
      by_vars = choices_selected(
        choices = variable_choices(ADLB, c("PARAM", "AVISIT")),
        selected = c("AVISIT")
      ),
      summarize_vars = choices_selected(
        choices = variable_choices(ADLB, c("AVAL", "CHG")),
        selected = c("AVAL")
      ),
      useNA = "ifany",
      paramcd = choices_selected(
        choices = value_choices(ADLB, "PARAMCD", "PARAM"),
        selected = "ALT"
      )
    )
  )
)
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 = ifelse(inherits(arm_var, "data_extract_spec"),
    teal.transform::datanames_input(arm_var), "ADSL"),
  arm_var,
  arm_ref_comp = NULL,
  paramcd,
  strata_var,
  aval_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "AVAL"), "AVAL", fixed = TRUE),
  cnsr_var = teal.transform::choices_selected(teal.transform::variable_choices(dataname,
    "CNSR"), "CNSR", fixed = TRUE),
  conf_level_coxph = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95, keep_order
    = TRUE),
  conf_level_survfit = teal.transform::choices_selected(c(0.95, 0.9, 0.8), 0.95,
    keep_order = TRUE),
  time_points,
  time_unit_var =
    teal.transform::choices_selected(teal.transform::variable_choices(dataname, "AVALU"),
      "AVALU", fixed = TRUE),
  event_desc_var = teal.transform::choices_selected("EVNTDESC", "EVNTDESC", fixed = TRUE),
  add_total = FALSE,
  total_label = default_total_label(),
  na_level = default_na_str(),
  pre_output = NULL,
  post_output = NULL,
  basic_table_args = teal.widgets::basic_table_args()
)
```

**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	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for variable names that can be used as arm_var. It defines the grouping variable(s) in the results table. If there are two elements selected for arm_var, second variable will be nested under the first variable.
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 <a href="#">teal.transform::variable_choices()</a> or delayed <a href="#">teal.transform::value_choices()</a> with the elements named ref and comp that the defined the default reference and comparison arms when the arm variable is changed.
paramcd	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the parameter code variable from dataname.
strata_var	( <a href="#">teal.transform::choices_selected()</a> ) names of the variables for stratified analysis.
aval_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the analysis variable.
cnsr_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for the censoring variable.
conf_level_coxph	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for confidence level, each within range of (0, 1).
conf_level_survfit	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for confidence level, each within range of (0, 1).
time_points	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and preselected option for time points that can be used in <a href="#">tern::surv_timepoint()</a> .
time_unit_var	( <a href="#">teal.transform::choices_selected()</a> ) object with all available choices and pre-selected option for the time unit variable.
event_desc_var	(character or <a href="#">data_extract_spec()</a> ) variable name with the event description information, optional.
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 <code>set_default_total_label("new_default")</code> .

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 <code>set_default_na_str("new_default")</code> .
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 <code>shiny::helpText()</code> elements are useful.
basic_table_args	(basic_table_args) optional object created by <code>teal.widgets::basic_table_args()</code> with settings for the module table. The argument is merged with option <code>teal.basic_table_args</code> and with default module arguments (hard coded in the module body). For more details, see the vignette: <code>vignette("custom-basic-table-arguments", package = "teal.widgets")</code> .

## Details

- The core functionality of this module is based on `coxph_pairwise()`, `surv_timepoint()`, and `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.

## See Also

The [TLG Catalog](#) where additional example apps implementing this module can be found.

## Examples

```
ADSL <- tmc_ex_adsl
ADTTE <- tmc_ex_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 = cdisc_data(
    ADSL = ADSL,
    ADTTE = ADTTE,
    code = "
      ADSL <- tmc_ex_adsl
      ADTTE <- tmc_ex_adtte
    "
  ),
  modules = modules(
    tm_t_tte(
      label = "Time To Event Table",
      dataname = "ADTTE",
      arm_var = choices_selected(
        variable_choices(ADSL, c("ARM", "ARMCD", "ACTARMCD")),
        "ARM"
      ),
      arm_ref_comp = arm_ref_comp,
      paramcd = choices_selected(
        value_choices(ADTTE, "PARAMCD", "PARAM"),
        "OS"
      ),
      strata_var = choices_selected(
        variable_choices(ADSL, c("SEX", "BMRKR2")),
        "SEX"
      ),
      time_points = choices_selected(c(182, 243), 182),
      event_desc_var = choices_selected(
        variable_choices(ADTTE, "EVNTDESC"),
        "EVNTDESC",
        fixed = TRUE
      )
    )
  )
)
if (interactive()) {
  shinyApp(app$ui, app$server)
}

```

# Index

- \* **datasets**
  - ex\_data, [15](#)
- add\_expr, [5](#)
- as\_num, [6](#)
- base::factor(), [81](#)
- base::Paren(), [7](#), [21](#)
- bracket\_expr, [7](#)
- call\_concatenate, [8](#)
- choices\_selected(), [22](#)
- clean\_description, [9](#)
- color\_lab\_values, [9](#)
- column\_annotation\_label, [10](#)
- control\_coxph\_annot(), [50](#)
- control\_surv\_med\_annot(), [50](#)
- coxph\_pairwise(), [139](#)
- cs\_to\_des\_filter, [11](#)
- cs\_to\_des\_select, [12](#)
- cs\_to\_filter\_spec, [12](#)
- cs\_to\_select\_spec, [13](#)
- data\_extract\_spec(), [138](#)
- default\_total\_label, [14](#)
- ex\_data, [15](#)
- extract\_input, [15](#)
- get\_var\_labels, [17](#)
- ggplot2::ggplot(), [30](#), [34](#), [45](#), [52](#), [56](#), [59](#), [63](#), [66](#)
- h\_concat\_expr, [18](#)
- 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](#), [29](#), [31](#), [35](#), [39](#), [43](#), [47](#), [51](#), [54](#), [57](#), [61](#), [64](#), [67](#), [71](#), [74](#), [77](#), [81](#), [85](#), [90](#), [92](#), [95](#), [99](#), [103](#), [105](#), [109](#), [110](#), [112](#), [115](#), [117](#), [120](#), [123](#), [126](#), [129](#), [132](#), [135](#), [139](#)
- shiny::selectInput(), [11–13](#)
- split\_choices, [22](#)
- split\_col\_expr, [23](#)
- split\_interactions, [24](#)
- surv\_time(), [139](#)
- surv\_timepoint(), [139](#)
- teal.modules.clinical  
(teal.modules.clinical-package), [4](#)
- teal.modules.clinical-package, [4](#)
- teal.transform::choices\_selected(), [11–13](#), [18](#), [25](#), [28](#), [35](#), [38](#), [42](#), [43](#), [46](#), [49](#), [50](#), [53](#), [56](#), [57](#), [60](#), [63](#), [64](#), [67](#), [70](#), [73](#), [74](#), [76](#), [77](#), [80](#), [84](#), [89](#), [92](#), [95](#), [98](#), [102](#), [105](#), [108](#), [110](#), [112](#), [114](#), [116](#), [119](#), [122](#), [125](#), [128](#), [129](#), [131](#), [134](#), [138](#)
- teal.transform::data\_extract\_spec(), [11](#), [12](#)
- teal.transform::filter\_spec(), [13](#)
- teal.transform::value\_choices(), [25](#), [28](#), [38](#), [42](#), [50](#), [76](#), [80](#), [84](#), [105](#), [138](#)
- teal.transform::variable\_choices(), [25](#), [28](#), [38](#), [42](#), [50](#), [76](#), [80](#), [84](#), [105](#), [138](#)
- teal.widgets::basic\_table\_args(), [26](#), [29](#), [71](#), [74](#), [77](#), [81](#), [85](#), [90](#), [93](#), [96](#), [99](#), [103](#), [106](#), [109](#), [120](#), [123](#), [126](#), [129](#), [132](#), [135](#), [139](#)
- teal.widgets::ggplot2\_args(), [29](#), [31](#), [35](#), [39](#), [43](#), [47](#), [54](#), [57](#), [61](#), [64](#), [68](#)
- tern::estimate\_odds\_ratio(), [80](#)

tern::estimate\_proportion\_diff(), 80  
tern::h\_stack\_by\_baskets(), 129  
tern::surv\_timepoint(), 138  
tern::test\_proportion\_diff(), 80  
tm\_a\_gee, 24  
tm\_a\_mmrn, 27  
tm\_g\_barchart\_simple, 30  
tm\_g\_ci, 34  
tm\_g\_forest\_rsp, 37  
tm\_g\_forest\_tte, 41  
tm\_g\_ipp, 45  
tm\_g\_km, 48  
tm\_g\_lineplot, 52  
tm\_g\_pp\_adverse\_events, 56  
tm\_g\_pp\_patient\_timeline, 59  
tm\_g\_pp\_therapy, 63  
tm\_g\_pp\_vitals, 66  
tm\_t\_abnormality, 69  
tm\_t\_abnormality\_by\_worst\_grade, 72  
tm\_t\_ancova, 75  
tm\_t\_binary\_outcome, 79  
tm\_t\_coxreg, 83  
tm\_t\_events, 88  
tm\_t\_events\_by\_grade, 91  
tm\_t\_events\_patyear, 94  
tm\_t\_events\_summary, 97  
tm\_t\_exposure, 101  
tm\_t\_logistic, 104  
tm\_t\_mult\_events, 107  
tm\_t\_pp\_basic\_info, 110  
tm\_t\_pp\_laboratory, 111  
tm\_t\_pp\_medical\_history, 114  
tm\_t\_pp\_prior\_medication, 116  
tm\_t\_shift\_by\_arm, 118  
tm\_t\_shift\_by\_arm\_by\_worst, 121  
tm\_t\_shift\_by\_grade, 124  
tm\_t\_smq, 127  
tm\_t\_summary, 131  
tm\_t\_summary\_by, 133  
tm\_t\_tte, 137  
tmc\_ex\_adae (ex\_data), 15  
tmc\_ex\_adaette (ex\_data), 15  
tmc\_ex\_adcm (ex\_data), 15  
tmc\_ex\_adeq (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\_adv (ex\_data), 15