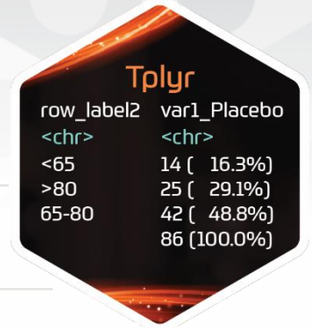
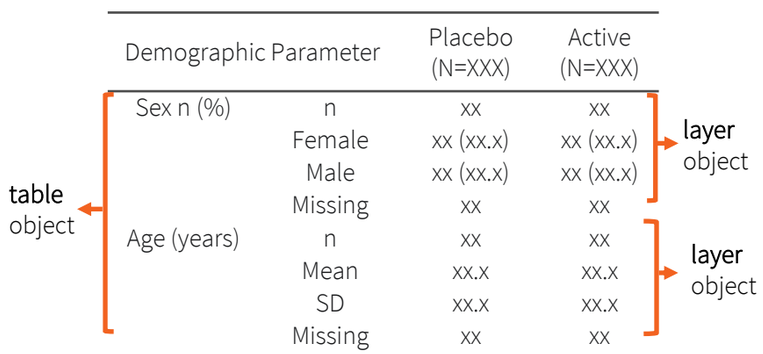
Building Clinical Safety Summaries with Tplyr: : CHEAT SHEET



'Tplyr' contains intuitive functions that build upon one another to create summary tables, which eliminates the redundancy of programming all while remaining flexible enough to conform to varying standards.



CREATING THE TABLE OBJECT

tplyr_table(target, treat_var, where=TRUE, cols=vars()) – used to create the table object

Parameter	Description
target	dataset used to perform summaries
treat_var	variable used to distinguish treatment groups
where=	subset applied to table level
cols=	grouping variable(s) used to create columns on the display (Note: this is in addition to treat_var)

CREATING LAYER OBJECTS

group_<type>(parent, target_var, by=vars(), where=TRUE, …) - family of functions used to create layers.

Parameter	Description
parent	the tplyr_table() object
target_var	variable(s) on which the summary is performed
by=	variable(s) or value(s) used as grouping variable(s) and represented as row label(s)
where=	subset applied to layer level (Note: this is in addition to any subset applied at the table level)

HELPER FUNCTIONS

To get the **underlying raw calculations**:

get_numeric_data(x, layer=NULL, where=TRUE, …) – Provides access to the un-formatted numeric data for each layer. *get_numeric_data(t)*

get_stats_data(x, layer = NULL, statistic = NULL, where = TRUE, ...) – Provides access to the un-formatted numeric statistical data for each layer. *get_stats_data(t)*

LAYER TYPES

COUNT LAYERS

group_count() - Specifies that a layer will be created to count occurrences and/or their proportions. *group_count(t, SEX, by="Sex n (%)")*

Sex n (%)	F	53 (61.6%)
	M	33 (38.4%)

SHIFT LAYERS

group_shift() - Specifies a shift layer will be created to count occurrences and their proportions from one state to another. *group_shift(t, vars(row=BNRIND, column=ANRIND), by=vars(PARAM,AVISIT))*

PARAM	VISIT		L	N	H
PARAM 1	VISIT 1	L	0	0	1
		N	3	12	0
		H	0	7	2

DESCRIPTIVE STATISTICS LAYERS

group_desc() - Specifies a layer will be created to perform summaries on continuous variables. *group_desc(t, AGE, by="Age (years)"))*

Age (years)	n	86
	Mean (SD)	75.2 (8.59)
	Median	76.0
	Q1, Q3	69.2, 81.8
	Min, Max	52, 89
	Missing	0

ADDING LAYERS TO A TABLE

add_layer(parent, layer, name=NULL) Constructs the layer within the call to the function.

add_layers(parent, …) Attaches layers that have already been constructed.

Parameter	Description
parent	the tplyr_table() object
layer	contains the group_type() function call and any modifier functions to create the layer
name=	specifies the layers name within the tplyr_table() object's layer container
…	specifies the layer objects that will be attached to the tplyr_table() object

PROCESSING THE DATA

Constructing a table or layer object constructs the metadata necessary to generate a table but does not process the actual data. To generate the data and perform the summaries use the **build**() function.

SORTING

Ordering helpers are columns added into 'Tplyr' tables.

SORTING THE LAYERS

Layers are indexed using the variable ord_layer_index by the order in which they were added to the table using add_layer() or add_layers().

SORTING THE BY VARIABLES

Each by variable gets an ord_layer_<n> column. The order variables will calculate based on the first applicable method:

- Use factor levels if variable is a factor
- Use a matching variable name suffixed by *N* from the dataset if available (i.e. RACE and RACE*N*)
- Use alphanumeric sorting of variable values

SORTING DESCRIPTIVE STATISTICS LAYER RESULTS

Descriptive statistics layers get an ord_layer_<n> column based on the order in which the f_str() objects are created through set_format_strings().

SORTING COUNT LAYER RESULTS

Count layers get an ord_layer_<n> column based on the sort method specified in **set_order_count_method()**.

set_order_count_method("byfactor") - Use factor levels. If variable is not a factor, alphanumeric sorting will be used. This is the default method and set_order_count_method() does not need to be called.

set_order_count_method("byvarn") - Use a matching variable name suffixed by *N* from the dataset if available (i.e. RACE and RACE*N*)

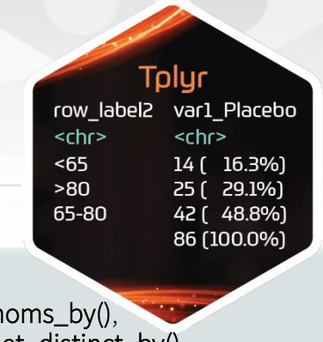
set_order_count_method("bycount") – Sort based on counts in a particular column. Requires the use of additional helper functions:

- **set_ordering_cols**(e, …) – Specifies the treat_var and cols= value(s) from tplyr_table() to determine the column from which the ordering should be based. *set_ordering_cols("High","WHITE")*
- **set_result_order_var**(e, result_order_var) – Specifies the occurrence or proportion variable on which the ordering should be based. *set_result_order_var(n)*

atorus
Your Complete Data Solution

 Atorus Research • Ashley.Tarasiewicz@atorusresearch.com • atorusresearch.com • Learn more at https://atorus-research.github.io/Tplyr/ • package version 0.3.1 • Updated: 2021-01

Building Clinical Safety Summaries with Tplyr: : CHEAT SHEET



COUNT AND SHIFT LAYERS

CALCULATING PERCENTAGES

set_denoms_by(e, …) - Specifies variable(s) to use to calculate percentages. If not called, uses treat_var and cols= from tplyr_table().

set_denom_where(e, denom_where) - Specifies denominator subset. If not called, uses where= from group_<type>().

MISSING COUNTS PRESENTATION

set_missing_count(e, fmt = NULL, sort_value = NULL, denom_ignore = FALSE, ...) – Controls how missing counts are handled.

ADDING A 'TOTAL' ROW

add_total_row(e, fmt = NULL, count_missings = TRUE, sort_value = NULL) - Adds a row presenting the total counts (i.e., the n's that are summarized).

set_total_row_label(e, total_row_label) - Specifies a row label for the total row. If not called, default text will be "Total".

NESTED COUNTS

When calculating **nested counts** use dplyr::vars() to specify 2 variables for target_var.

DISTINCT VS EVENT COUNTS

set_distinct_by(e, distinct_by) - Specifies variable(s) to use to calculate distinct occurrences.

FORMATTING

set_format_strings() and **f_str()** are used to specify the occurrence and proportion variables and how they will be presented. The user uses x's to specify how the numbers will be displayed.

```
t <- tplyr_table(adsl, TRT01P, where=SAFFL=="Y") %>%
  add_total_group() %>%
  add_treat_grps('Treated' = c("Xanomeline High Dose", "Xanomeline Low Dose")) %>%
  add_layer(
    group_count(AGEGR1, by= RACE, where=SEX=="F") %>%
      set_denoms_by(TRT01P, RACE) %>%
      set_denom_where(TRUE) %>%
      set_missing_count(f_str("xx", n), Missing=NA, denom_ignore=TRUE) %>%
      add_total_row(fmt=f_str("xx", n), count_missings=FALSE) %>%
      set_total_row_label("n")
  )
t %>%
  build()
```

```
t <- tplyr_table(adae, TRTA, where=AESER=="Y") %>%
  set_pop_data(adsl) %>%
  set_pop_treat_var(TRT01A) %>%
  set_pop_where(SAFFL=="Y") %>%
  add_layer(
    group_count(vars(AEBODSYS,AEDECOD)) %>%
      set_distinct_by(USUBJID) %>%
      set_format_strings(f_str("xx (xx.x) [xx]", distinct_n, distinct_pct, n))
  )
t %>%
  build()
```

group_shift() can be used with the **set_denoms_by()**, **set_denom_where()**, **set_distinct_by()**, **set_format_strings()**, and **f_str()** functions.

TABLE LEVEL FUNCTIONS

ADDING TREATMENT GROUPS

add_treat_grps(table, …) – Create new treatment groups by combining existing treatment groups from the values within treat_var.

add_total_group(table, group_name="Total") - Create total treatment group by combining all treatment groups from the values within treat_var.

ADDING A POPULATION DATASET

If target does not include the entire necessary population, the **population functions** can provide population information.

set_pop_data(table, pop_data) - Specifies a population dataset

set_pop_treat_var(table, pop_treat_var) - Specifies a treatment variable from the population dataset. If not called, uses treat_var from tplyr_table().

set_pop_where(obj, where) - Specifies a population subset. If not called, uses where= from tplyr_table().

DESCRIPTIVE STATISTIC LAYERS

BUILT-IN SUMMARIES

Description	Variable Name
N	n
Mean	mean
Standard Deviation	sd
Median	median
Variance	variance
Minimum	min
Maximum	max
Interquartile Range	iqr
Q1	q1
Q3	q3
Missing	missing

```
t <- tplyr_table(adsl, TRT01P, where=SAFFL=="Y") %>%
  add_layer(
    group_desc(BMIBL, by = "BMI at Baseline") %>%
      set_custom_summaries(geometric_mean = exp(sum(log(.var[.var > 0]), na.rm=TRUE) / length(.var))) %>%
      set_format_strings(
        "N"                    = f_str("xx", n),
        "Geometric Mean (SD)" = f_str("xx.a+1 (xx.a+2)", geometric_mean, sd, empty="NA"),
        cap=c(int=3,dec=2)
      )
  )
t %>%
  build()
```

CUSTOM SUMMARIES

Custom summaries allow any function to be used in a descriptive statistics layer.

set_custom_summaries(e, …) – Allows user to define custom summaries that will be performed in dplyr::summarize. Use .var as the variable name being summarized.

FORMATTING AND PERFORMING SUMMARIES

set_format_strings() and **f_str()** are used to specify the summaries that will be performed and how they will be presented.

- On the left side of the equal sign the user inputs text that becomes the row label.
- On the right side the user uses x's to specify how the numbers will be displayed and lists the descriptive statistic summaries that will be performed.

The empty parameter of **f_str()** specifies what to display if an element or elements in a cell produce NA values.

Auto precision is used to format numeric summaries based on the precision of the data collected.

- Use a instead of x (only 1 a is needed on each side of the decimal)
- Use a+n where n is the number of additional spaces you wish to add
- Use the cap parameter to cap the length allotted for integers and decimals

atorus
Your Complete Data Solution