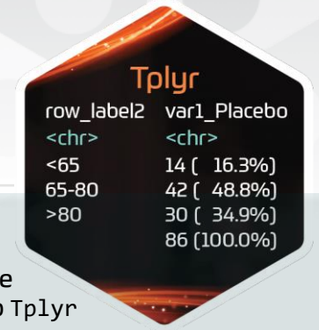
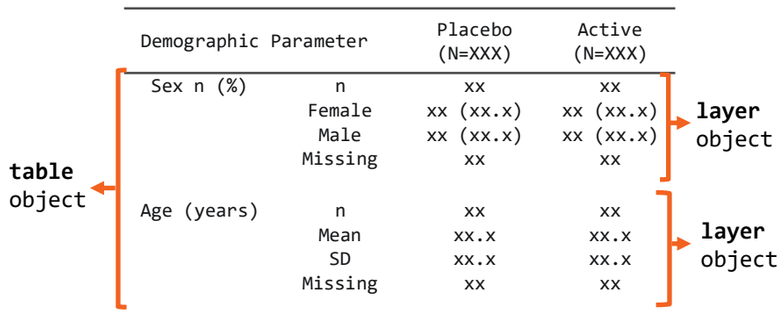
Traceability Focused Clinical Data Summary with Tplyr: : CHEAT SHEET





CREATING THE TABLE OBJECT

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

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.

CREATING LAYER OBJECTS

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

group_count(t, SEX, by="Sex n (%)")

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

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	1	0
		H	0	7	2

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

BUILDING AND USING METADATA

build(x, metadata) – Triggers the execution of the tplyr_table and optionally the associated metadata.

```
t <- tplyr_table(adsl, TRT01P, where = SAFFL == "Y") %>%
  add_layer(group_count(RACE))
t %>%
  build(metadata = TRUE)
```

row_id	row_label1	var1_Placebo	var1_Treated
c1_1	ASIAN	0 (0.0%)	2 (20.0%)
c2_1	WHITE	9 (100.0%)	8 (80.0%)

get_meta_subset(x, row_id, column, add_cols=vars(USUBJID), ...) – Extracts the subset of data based on result metadata.

```
get_meta_subset(t, 'c1_1', 'var1_Treated', add_cols = vars(USUBJID, SEX))
```

USUBJID	SEX	TRT01P	SAFFL	RACE
004	F	Treated	Y	ASIAN
007	M	Treated	Y	ASIAN

get_meta_result(x, row_id, column, ...) – Extracts the result metadata of a tplyr_table.

```
get_meta_result(t, 'c1_1', 'var1_Treated')
#> tplyr_meta: 3 names, 3 filters
#> Names:
#>     TRT01P, SAFFL, RACE
#> Filters:
#>     TRT01P == c("Treated"), SAFFL == "Y", RACE == c("ASIAN")
```

TEMPLATES AND TABLE FORMATS

new_layer_template(name, template) – Creates a layer template.

use_template(name, ..., add_params=NULL) – Uses a layer template.

set_<type>_layer_formats(obj, ...) – Sets default format strings for layers type.

GENERAL STRING FORMATTING

The f_str() object controls the numbers reported.

xx (xx.x%)

8 (53.3%)
1 (6.7%)

Variable 1:
- integer with 2 spaces

Variable 2:
- integer with 2 spaces and 1 decimal place
- inside parentheses, appended with the % symbol

xx (XX.x%)

8 (53.3%)
1 (6.7%)

Variable 1:
- integer with 2 spaces

Variable 2:
- integer with 2 spaces and 1 decimal place
- inside parentheses, appended with the % symbol
- "hug" left parenthesis indicated by uppercase letter(s)

- Decimals round to the specified length.
- Integers will not truncate. If an integer exceeds the set length, Tplyr will push the number over.

POST PROCESSING

str_indent_wrap(x, width=10, tab_width=5) – Wrap strings to a specific width with hyphenation while preserving indentation.

apply_row_masks(dat, row_breaks=FALSE, ...) – Replace repeating row label variables with blanks in preparation for display and optionally inserts row breaks.

apply_conditional_format(string, format_group, condition, replacement, full_string=FALSE) – Applies conditional formatting of a pre-populated string of numbers.

i.e. "0 (0.0%)" -> "0" or "1 (0.004%)" -> "1 (<0.1%)"

str_extract_fmt_group(string, format_group) – Extracts format group strings.

```
string <- c(" 5  (5.8%)", " 8  (9.3%)", "78 (90.7%)")
str_extract_fmt_group(string, 1)
#> [1] " 5" " 8" "78"
str_extract_fmt_group(string, 2)
#> [1] "(5.8%)"  "(9.3%)"  "(90.7%)"
```

str_extract_num(string, format_group) – Extracts format group numbers.

apply_formats(format_string, ..., empty=c(.overall="")) – Applies format strings outside of a tplyr_table.

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 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*)

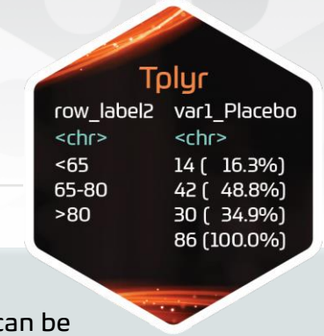
set_order_count_method("bycount") – Sort based on counts in a particular column. Requires the use of additional helper functions:

- **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)*

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().

Traceability Focused Clinical Data Summary 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.

```
t <- tplyr_table(adsl, TRT01P, where = SAFFL == "Y") %>%
   add_total_group() %>%
   add_treat_grps('Treated' = c("High Dose", "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(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(TRTA) %>%
   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() is an abstraction of **group_count()** and can be used with many of the same 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 specifies 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/A instead of x/X
- Use a+n/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