**Solutions – Beginner**

1. **3 Result columns –** These are based on the available treatment groups. The treatment groups are determined by the **treat\_var** parameter of **tplyr\_table.** In a built tplyr\_table, the result columns will always start with **var**. You can view this information by printing the dataset

# call layout lyt

t1 <- tplyr\_table(advs, ARM) %>%

set\_pop\_data(adsl) %>%

set\_pop\_treat\_var(ARM) %>%

add\_layer(

group\_desc(AVAL, by = vars(AVISIT, VSTEST))

) %>%

dat <- t1 %>%

build()

dat

1. **ADSL –** The Tplyr function **set\_pop\_data()** specifies which dataset should be treated as the population dataset, and header N counts will be derived from this source. Printing a tplyr\_table will show you this information.

# call layout lyt

> t1

\*\*\* tplyr\_table \*\*\*

Target (data.frame):

Name: advs

Rows: 2930

Columns: 103

**pop\_data (data.frame)**

**Name: adsl**

**Rows: 254**

**Columns: 3**

treat\_var variable (quosure)

ARMpop\_treat\_var variable (quosure)

ARM

header\_n: 2 header groups

treat\_grps groupings (list)

Table Columns (cols):

where: TRUE

Number of layer(s): 1

layer\_output: 0

1. **set\_format\_strings() –** and even more specifically, **f\_str()** is where the individual summaries are specified. These two function work together to specify the row\_label used for a summary, the string formatting of the results, and the summaries that will be calculated. With intuitive row labels, you can determine this information from the resulting dataset. But you can also look inside the tplyr\_table object to check the format strings assigned to a layer. *Note that doing this requires a reasonable understanding of how to investigate objects in R and is not a basic skill.*

> t$layers[[1]]

\*\*\* desc\_layer \*\*\*

Self: desc\_layer < 0x55ca25a27798 >

Parent: tplyr\_table < 0x55ca229883d0 >

target\_var:

AVAL

by: VISIT VSTEST

where: TRUE

Layer(s): 0

> t$layers[[1]]$format\_strings

$n

\*\*\* Format String \*\*\*

xx

\*\*\* vars, extracted formats, and settings \*\*\*

n formated as: xx

integer length: 2

decimal length: 0

Total Format Size: 2

$`Mean (sd)`

\*\*\* Format String \*\*\*

xx.xx (x.xx)

\*\*\* vars, extracted formats, and settings \*\*\*

mean formated as: xx.xx

integer length: 2

decimal length: 2

sd formated as: x.xx

integer length: 1

decimal length: 2

Total Format Size: 12

$IQR

\*\*\* Format String \*\*\*

x.xx

\*\*\* vars, extracted formats, and settings \*\*\*

iqr formated as: x.xx

integer length: 1

decimal length: 2

Total Format Size: 4

$`min - max`

\*\*\* Format String \*\*\*

xx.xx - xx.xx

\*\*\* vars, extracted formats, and settings \*\*\*

min formated as: xx.xx

integer length: 2

decimal length: 2

max formated as: xx.xx

integer length: 2

decimal length: 2

Total Format Size: 13

1. **The appropriate code is as follows:**

t <- tplyr\_table(advs, ARM) %>%

set\_pop\_data(adsl) %>%

set\_pop\_treat\_var(ARM) %>%

add\_layer(

group\_desc(AVAL, by = vars(VISIT, VSTEST)) %>%

set\_format\_strings(

'n' = f\_str('xx', n, empty="NA"),

'Mean (sd)' = f\_str('xx.xx (x.xxx)', mean, sd, empty="NA"),

'IQR' = f\_str('x.xx', iqr, empty="NA"),

'min - max' = f\_str('xx.xx - xx.xx', min, max, empty='NA')

)

)

1. **They do!** Tplyr and rtables are both thoroughly tested packages. Think of this example in practice – realistically you could use one package to check the numbers from the other. Pretty cool!