Project: Chevron

A open-source package to facilitate generation of outputs

Liming Li, technical lead for chevron

29 Mar 2024 | Pharma RUG only

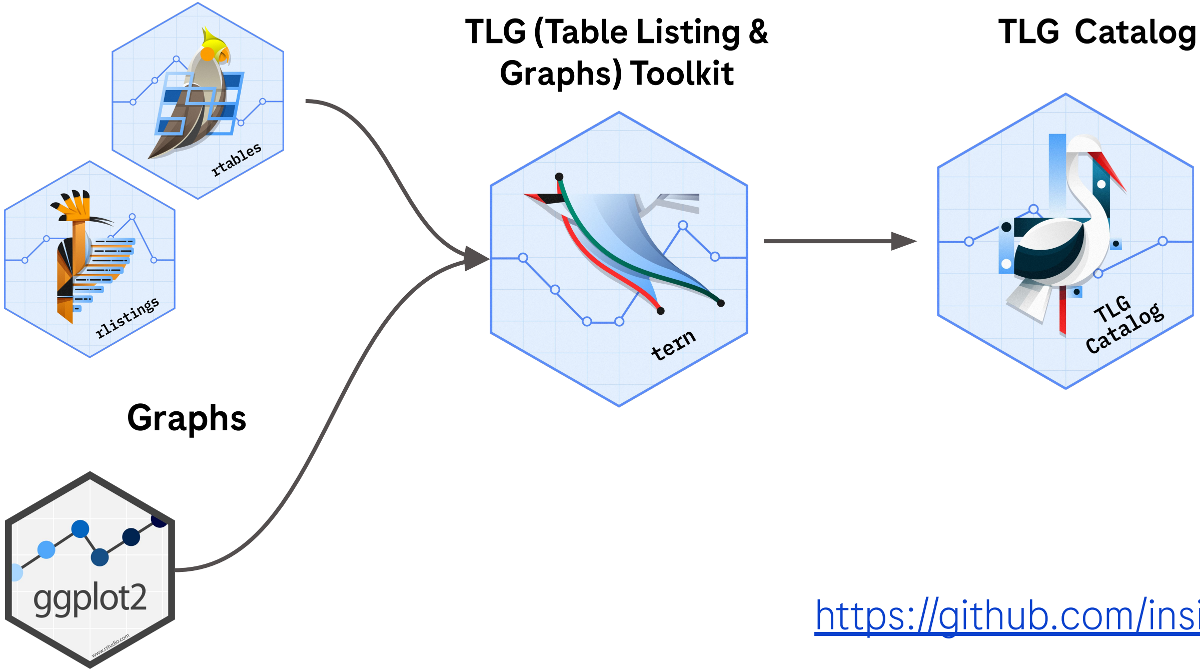
Table of contents

- Why we need new tools?
- The {chevron} package to fill the gap!

Why we need new tools?

The NEST packages

NEST is a collection of open-sourced R packages, which enables fast and efficient insights generation under clinical research settings, for both exploratory and regulatory purposes.



Tables & Listings

Graphs

TLG (Table Listing & Graphs) Toolkit

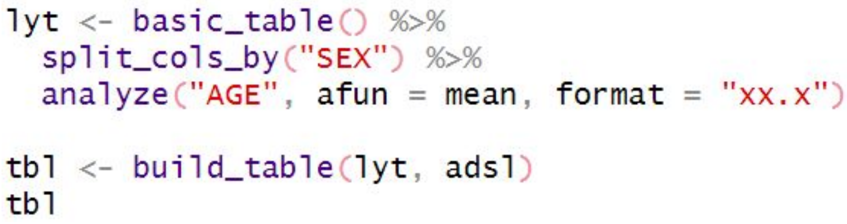
TLG Catalog

https://github.com/insightsengineering

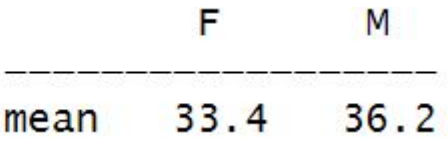
The NEST packages

Examples

{rtables}

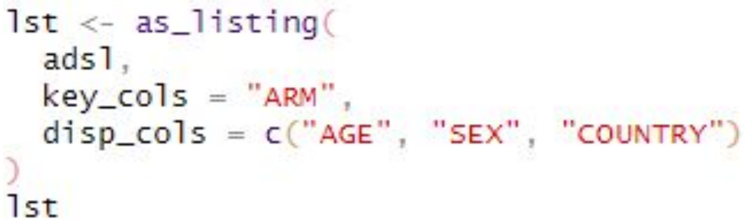


```r
lyt <- basic_table() %>%
  split_cols_by("SEX") %>%
  analyze("AGE", afun = mean, format = "xx.x")

tbl <- build_table(lyt, adsl)
tbl
```

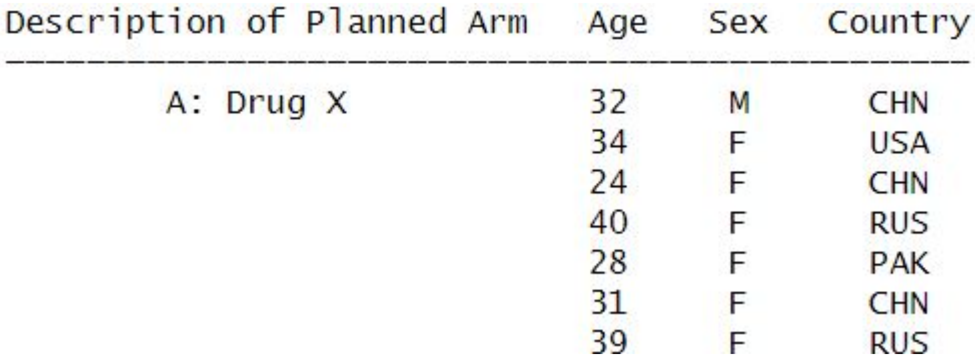


```
          F       M
———————————————————
mean    33.4    36.2
```

{rlistings}

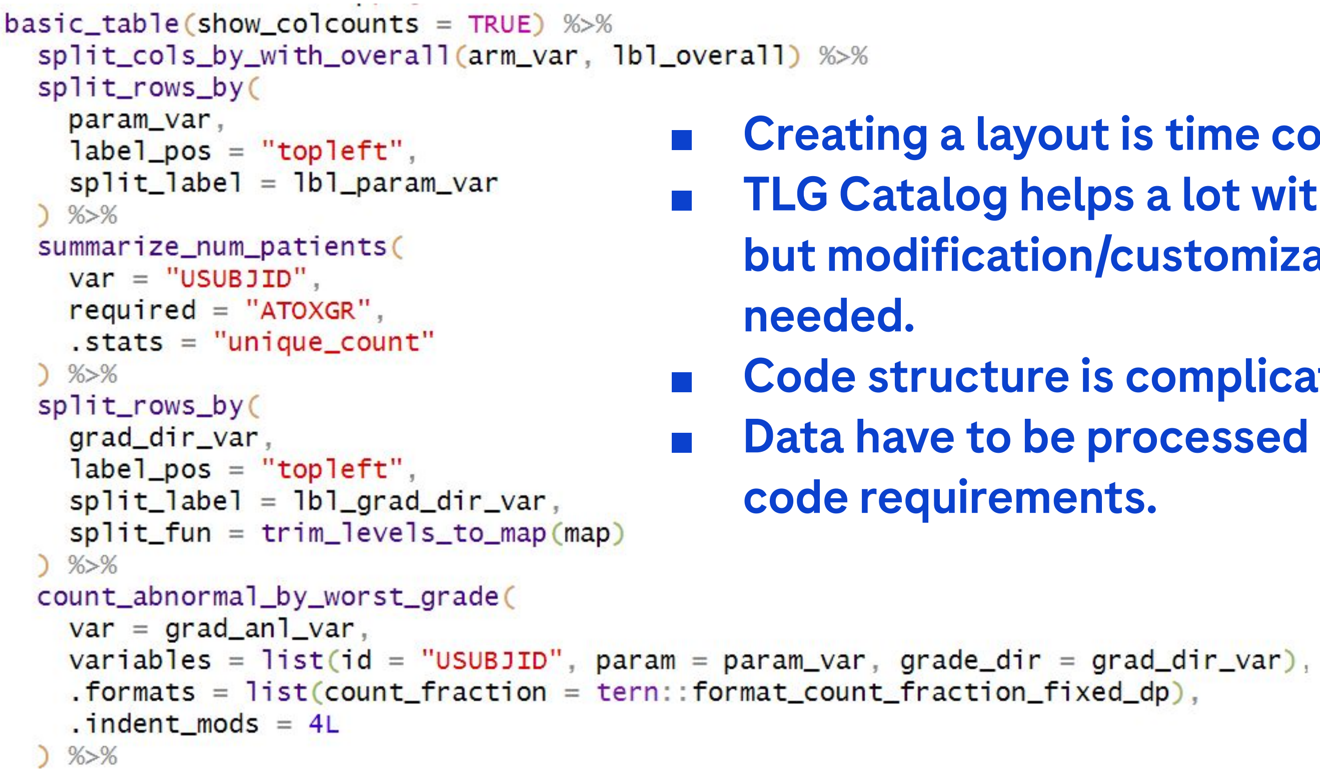


```r
lst <- as_listing(
  adsl,
  key_cols = "ARM",
  disp_cols = c("AGE", "SEX", "COUNTRY")
)
lst
```



```
Description of Planned Arm   Age   Sex   Country
————————————————————————————————————————————————
        A: Drug X            32     M      CHN
                             34     F      USA
                             24     F      CHN
                             40     F      RUS
                             28     F      PAK
                             31     F      CHN
                             39     F      RUS
```

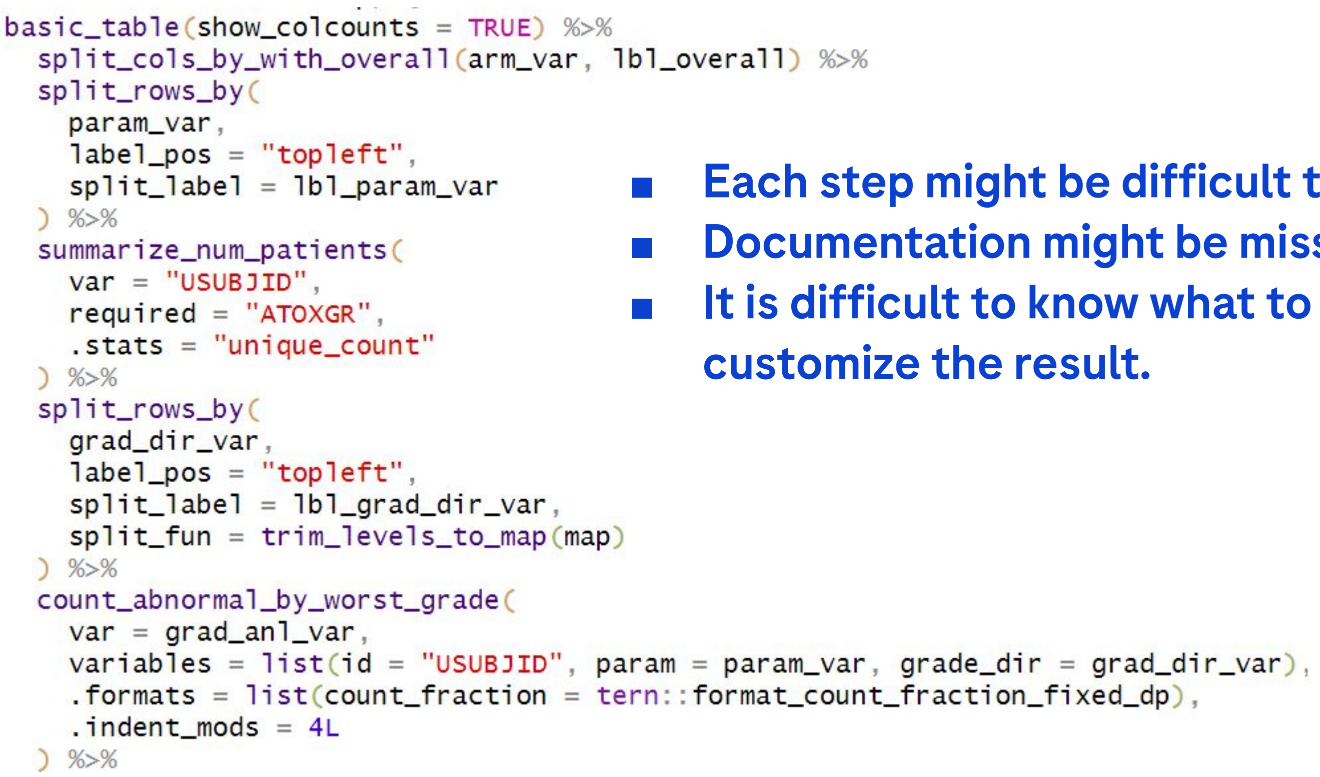
Creating TLG is (still) not easy



```
basic_table(show_colcounts = TRUE) %>%
  split_cols_by_with_overall(arm_var, lbl_overall) %>%
  split_rows_by(
    param_var,
    label_pos = "topleft",
    split_label = lbl_param_var
  ) %>%
  summarize_num_patients(
    var = "USUBJID",
    required = "ATOXGR",
    .stats = "unique_count"
  ) %>%
  split_rows_by(
    grad_dir_var,
    label_pos = "topleft",
    split_label = lbl_grad_dir_var,
    split_fun = trim_levels_to_map(map)
  ) %>%
  count_abnormal_by_worst_grade(
    var = grad_anl_var,
    variables = list(id = "USUBJID", param = param_var, grade_dir = grad_dir_var),
    .formats = list(count_fraction = tern::format_count_fraction_fixed_dp),
    .indent_mods = 4L
  ) %>%
```

- Creating a layout is time consuming.
- TLG Catalog helps a lot with a template, but modification/customization is usually needed.
- Code structure is complicated
- Data have to be processed to fit the code requirements.

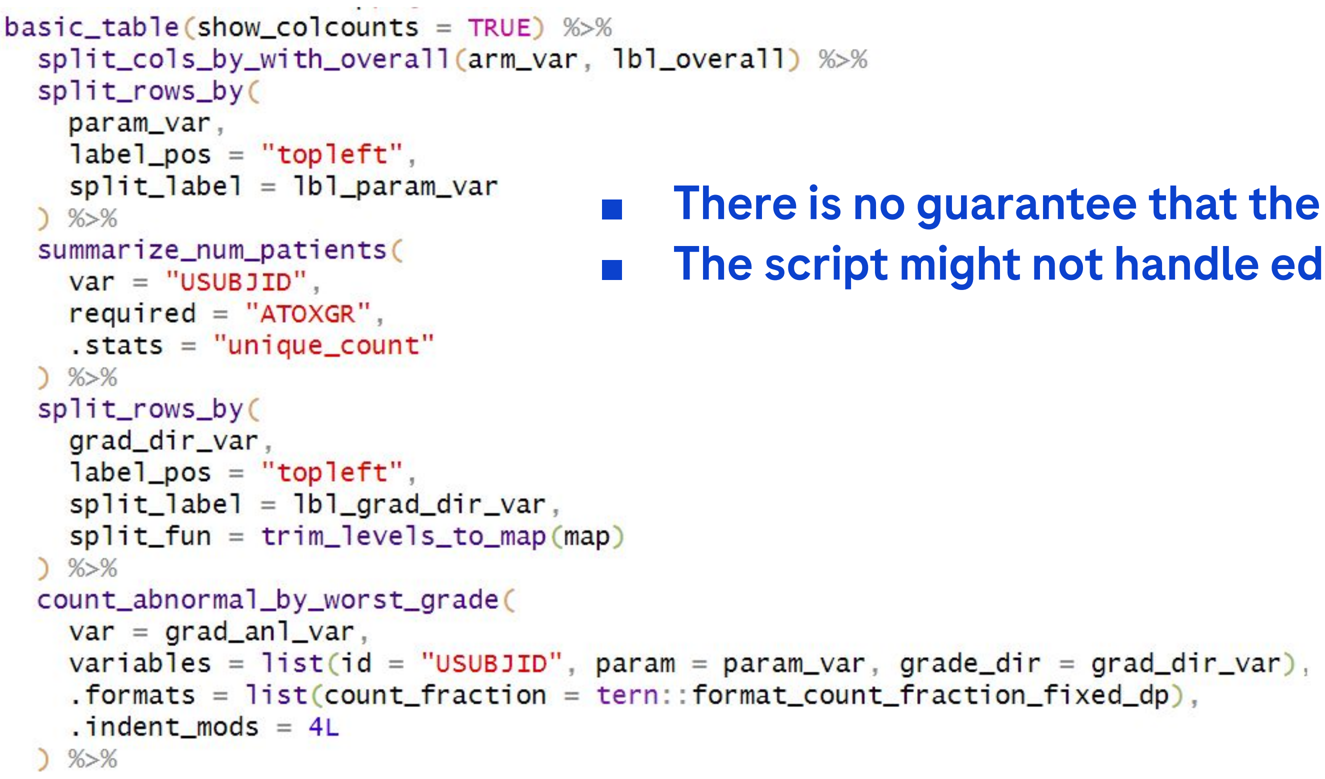
Custom code is not always documented



```
basic_table(show_colcounts = TRUE) %>%
  split_cols_by_with_overall(arm_var, lbl_overall) %>%
  split_rows_by(
    param_var,
    label_pos = "topleft",
    split_label = lbl_param_var
  ) %>%
  summarize_num_patients(
    var = "USUBJID",
    required = "ATOXGR",
    .stats = "unique_count"
  ) %>%
  split_rows_by(
    grad_dir_var,
    label_pos = "topleft",
    split_label = lbl_grad_dir_var,
    split_fun = trim_levels_to_map(map)
  ) %>%
  count_abnormal_by_worst_grade(
    var = grad_anl_var,
    variables = list(id = "USUBJID", param = param_var, grade_dir = grad_dir_var),
    .formats = list(count_fraction = tern::format_count_fraction_fixed_dp),
    .indent_mods = 4L
  ) %>%
```

- Each step might be difficult to understand
- Documentation might be missing in the script
- It is difficult to know what to modify to customize the result.

Script validation is difficult



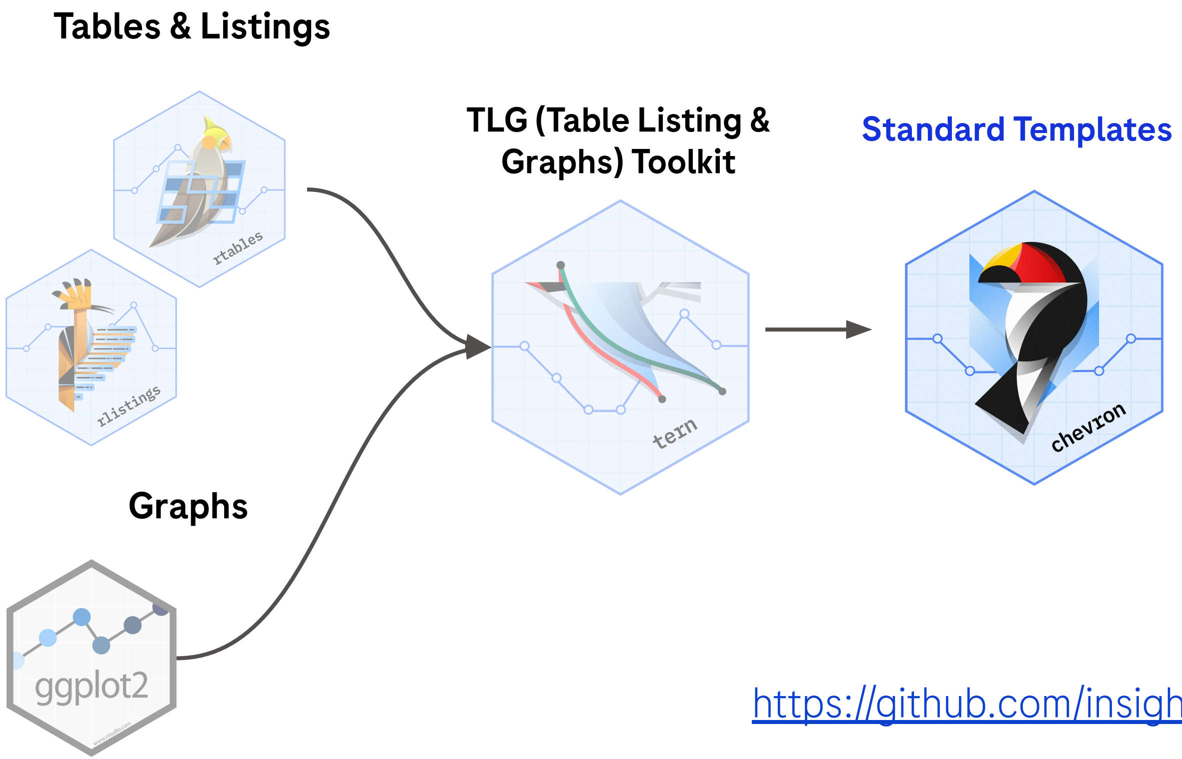
```
basic_table(show_colcounts = TRUE) %>%
  split_cols_by_with_overall(arm_var, lbl_overall) %>%
  split_rows_by(
    param_var,
    label_pos = "topleft",
    split_label = lbl_param_var
  ) %>%
  summarize_num_patients(
    var = "USUBJID",
    required = "ATOXGR",
    .stats = "unique_count"
  ) %>%
  split_rows_by(
    grad_dir_var,
    label_pos = "topleft",
    split_label = lbl_grad_dir_var,
    split_fun = trim_levels_to_map(map)
  ) %>%
  count_abnormal_by_worst_grade(
    var = grad_anl_var,
    variables = list(id = "USUBJID", param = param_var, grade_dir = grad_dir_var),
    .formats = list(count_fraction = tern::format_count_fraction_fixed_dp),
    .indent_mods = 4L
  ) %>%
```

- **There is no guarantee that the result is correct.**
- **The script might not handle edge cases well.**

The {chevron} package to fill the gap!

The {chevron} package

Chevron is built on top of the NEST family to provide robust and easy-to-use interfaces to facilitate the creation of standard output.



Tables & Listings

TLG (Table Listing & Graphs) Toolkit

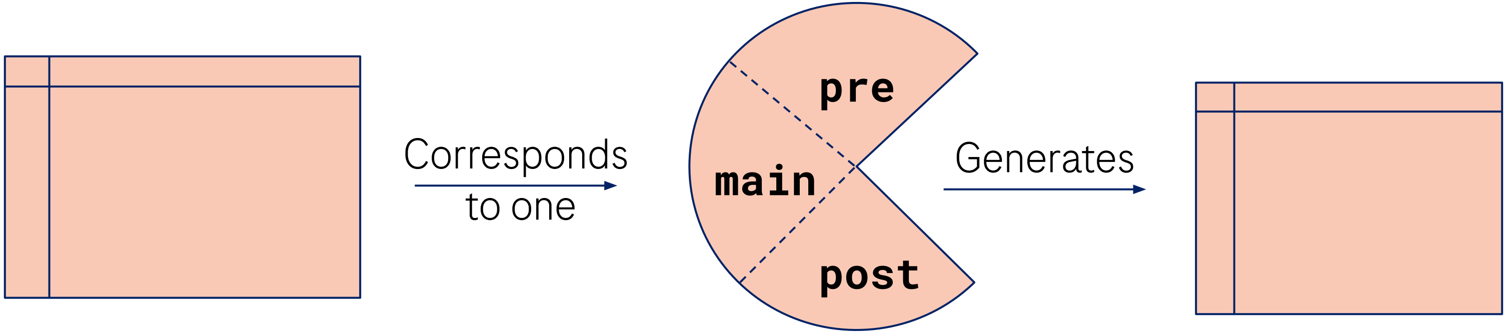
Standard Templates

Graphs

https://github.com/insightsengineering

The structure of {chevron} objects

{chevron} is a collection of high-level functions to create standard outputs for clinical trials reporting with limited parameterisation.



Corresponds to one

pre

main

post

Generates

Standard Template
Representing the structure of an expected table, listing or graph

chevron_tlg
An object with a preprocess, main and postprocess function

Table, listing or graph

The {chevron} package

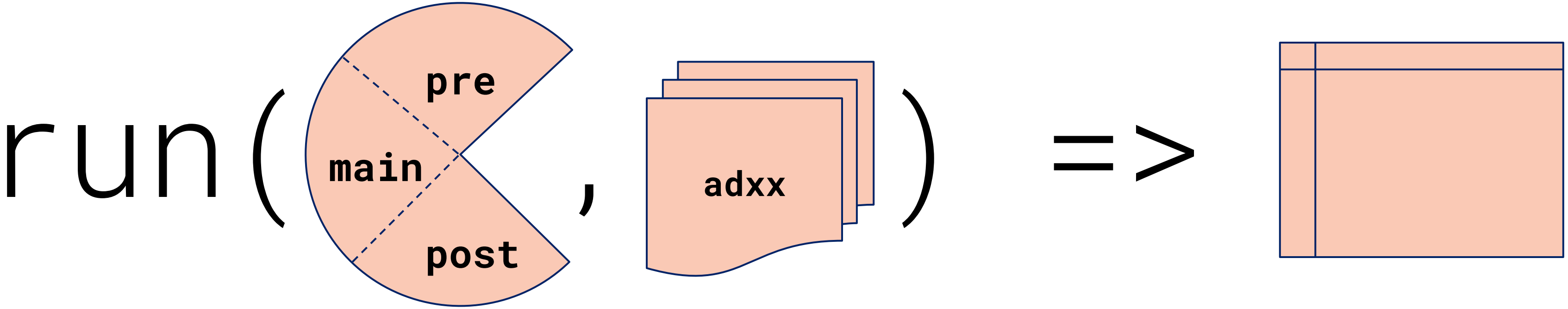
- **User Friendly**
  - Simple user interface
  - Informative log and error messages
  - Easy to control
  - Available on CRAN

- **Flexible & Scalable**
  - Easy to customize
  - Can be extended

- **Robust**
  - Daily validation
  - Extensive testing
  - Defensive programming

The `run` command

Create TLGs in one line



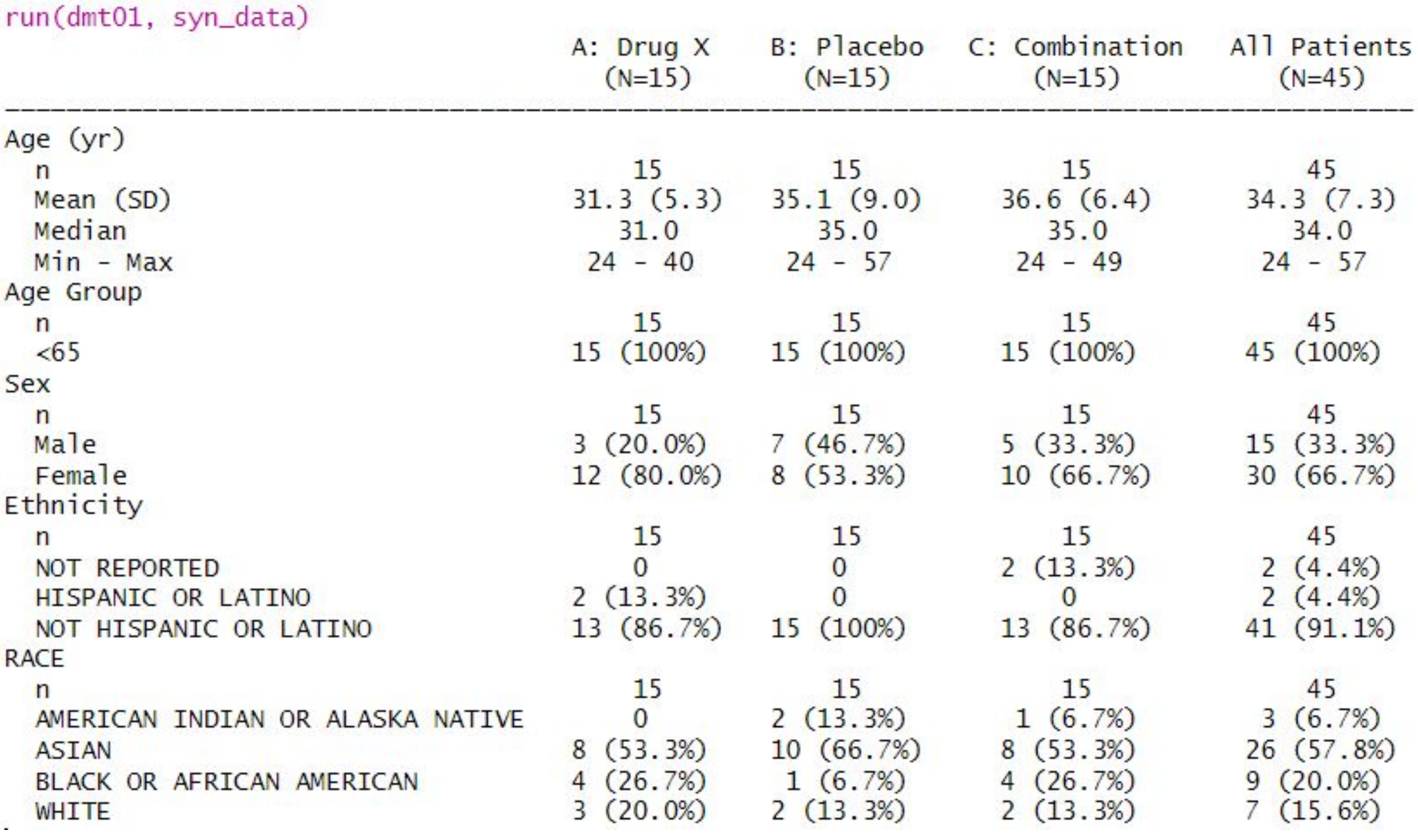
`chevron_tlg`
An object with a preprocess, main and postprocess function

`list of data.frames`
Representing an ADaM data set.

`Table, listing or graph`

The `run` command

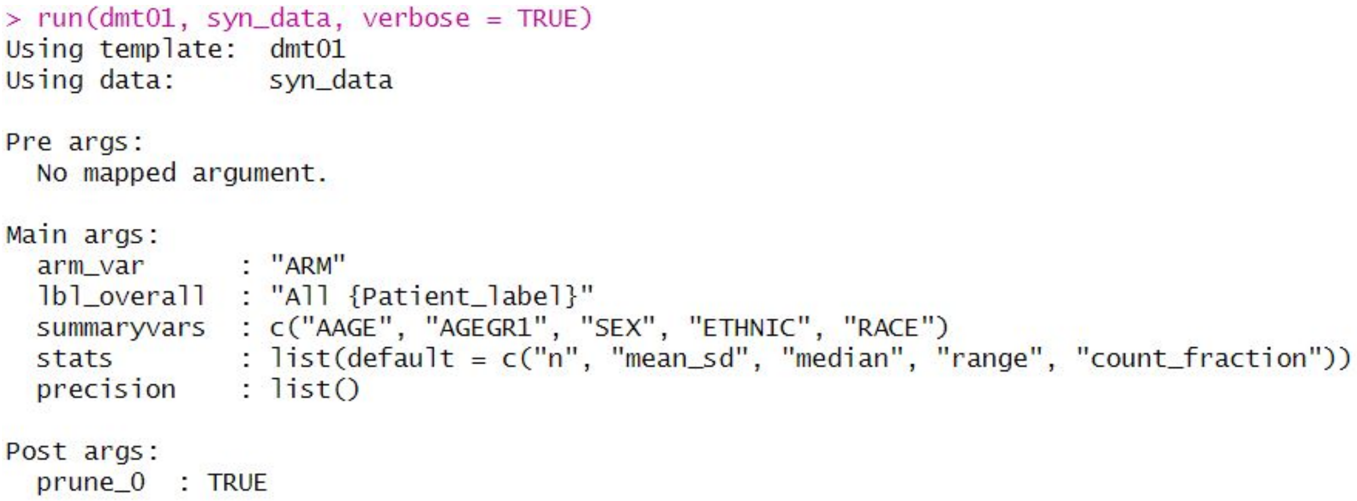
Create TLGs in one line: Example



```
run(dmt01, syn_data)
                                        A: Drug X    B: Placebo    C: Combination    All Patients
                                         (N=15)        (N=15)          (N=15)           (N=45)
--------------------------------------------------------------------------------------------------
Age (yr)
  n                                        15            15              15               45
  Mean (SD)                            31.3 (5.3)    35.1 (9.0)      36.6 (6.4)       34.3 (7.3)
  Median                                  31.0          35.0            35.0             34.0
  Min - Max                              24 - 40       24 - 57         24 - 49          24 - 57
Age Group
  n                                        15            15              15               45
  <65                                  15 (100%)     15 (100%)       15 (100%)        45 (100%)
Sex
  n                                        15            15              15               45
  Male                                 3 (20.0%)     7 (46.7%)       5 (33.3%)       15 (33.3%)
  Female                               12 (80.0%)    8 (53.3%)      10 (66.7%)       30 (66.7%)
Ethnicity
  n                                        15            15              15               45
  NOT REPORTED                              0             0          2 (13.3%)         2 (4.4%)
  HISPANIC OR LATINO                   2 (13.3%)          0               0            2 (4.4%)
  NOT HISPANIC OR LATINO               13 (86.7%)    15 (100%)      13 (86.7%)       41 (91.1%)
RACE
  n                                        15            15              15               45
  AMERICAN INDIAN OR ALASKA NATIVE          0        2 (13.3%)        1 (6.7%)         3 (6.7%)
  ASIAN                                8 (53.3%)     10 (66.7%)      8 (53.3%)       26 (57.8%)
  BLACK OR AFRICAN AMERICAN            4 (26.7%)      1 (6.7%)       4 (26.7%)        9 (20.0%)
  WHITE                                3 (20.0%)     2 (13.3%)       2 (13.3%)        7 (15.6%)
```

The run command

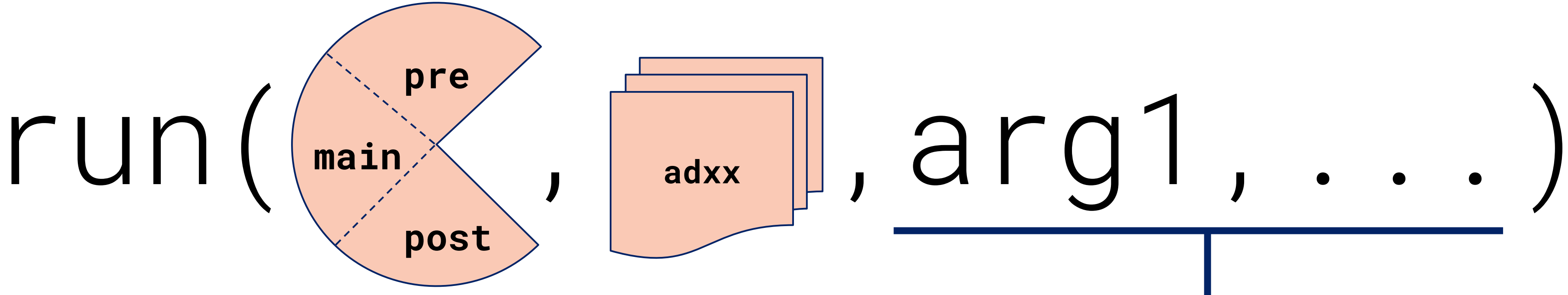
Generate log and create TLGs in one line: Example

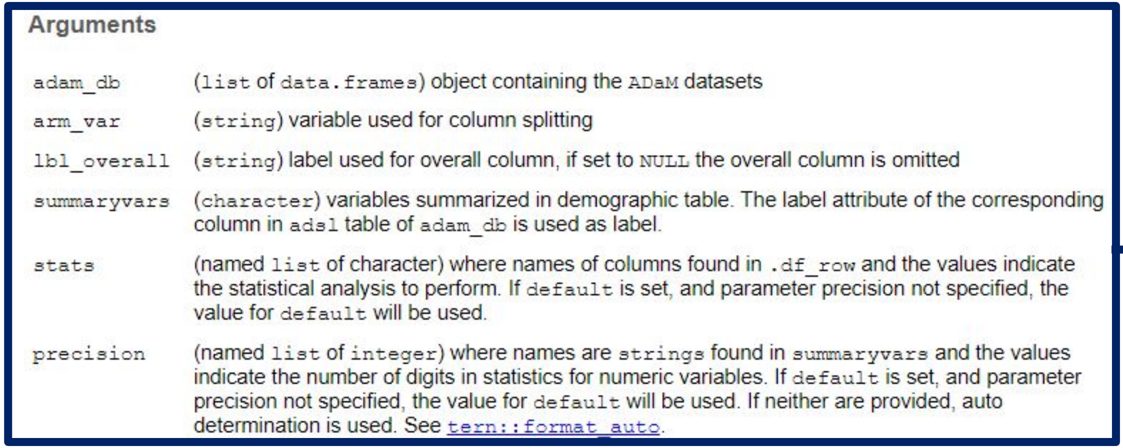


```
> run(dmt01, syn_data, verbose = TRUE)
Using template:  dmt01
Using data:      syn_data

Pre args:
  No mapped argument.

Main args:
  arm_var      : "ARM"
  lbl_overall  : "All {Patient_label}"
  summaryvars  : c("AAGE", "AGEGR1", "SEX", "ETHNIC", "RACE")
  stats        : list(default = c("n", "mean_sd", "median", "range", "count_fraction"))
  precision    : list()

Post args:
  prune_0  : TRUE
```

The controlling the `run` command

Customizing the output





Arguments

`adam_db` (`list` of `data.frames`) object containing the `ADaM` datasets

`arm_var` (`string`) variable used for column splitting

`lbl_overall` (`string`) label used for overall column, if set to `NULL` the overall column is omitted

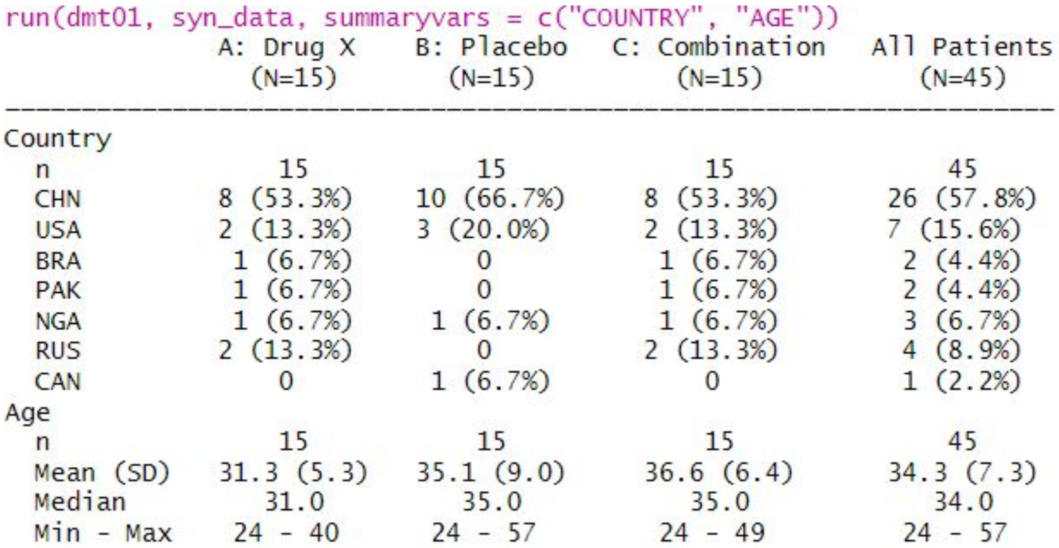
`summaryvars` (`character`) variables summarized in demographic table. The label attribute of the corresponding column in `adsl` table of `adam_db` is used as label.

`stats` (named `list` of character) where names of columns found in `.df_row` and the values indicate the statistical analysis to perform. If `default` is set, and parameter precision not specified, the value for `default` will be used.

`precision` (named `list` of `integer`) where names are `strings` found in `summaryvars` and the values indicate the number of digits in statistics for numeric variables. If `default` is set, and parameter precision not specified, the value for `default` will be used. If neither are provided, auto determination is used. See `tern::format_auto`.

The controlling the `run` command

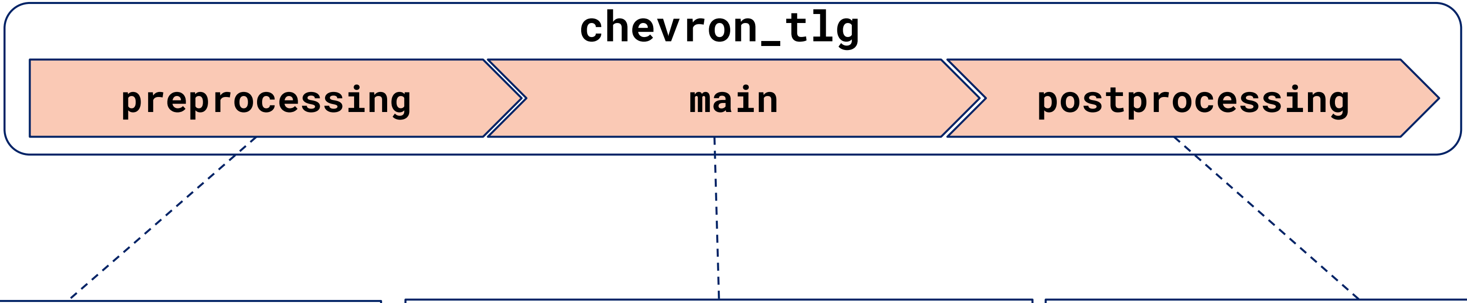
Customizing the output: Example



```
run(dmt01, syn_data, summaryvars = c("COUNTRY", "AGE"))
               A: Drug X     B: Placebo    C: Combination   All Patients
                (N=15)         (N=15)          (N=15)          (N=45)
-----------------------------------------------------------------------
Country
  n                15            15              15              45
  CHN          8 (53.3%)    10 (66.7%)      8 (53.3%)       26 (57.8%)
  USA          2 (13.3%)    3 (20.0%)       2 (13.3%)       7 (15.6%)
  BRA           1 (6.7%)        0            1 (6.7%)        2 (4.4%)
  PAK           1 (6.7%)        0            1 (6.7%)        2 (4.4%)
  NGA           1 (6.7%)     1 (6.7%)        1 (6.7%)        3 (6.7%)
  RUS          2 (13.3%)        0           2 (13.3%)        4 (8.9%)
  CAN              0         1 (6.7%)           0            1 (2.2%)
Age
  n                15            15              15              45
  Mean (SD)    31.3 (5.3)   35.1 (9.0)      36.6 (6.4)      34.3 (7.3)
  Median          31.0          35.0            35.0            34.0
  Min - Max     24 - 40       24 - 57         24 - 49         24 - 57
```

The chevron_tlg object

Internal structure



chevron_tlg

preprocessing → **main** → **postprocessing**

```
function(adam_db, …) {
   adam_db <-
wrangling(adam_db)
   return(adam_db)
}
```

Expect: ADaM dataset
Return: processed dataset

E.g.: subset rows, create new columns, drop levels

```
function(adam_db, …) {
   tlg <- build_tlg(adam_db)
   return(tlg)
}
```

Expect: processed adam dataset
Return: TLG

E.g.: Create a demography table

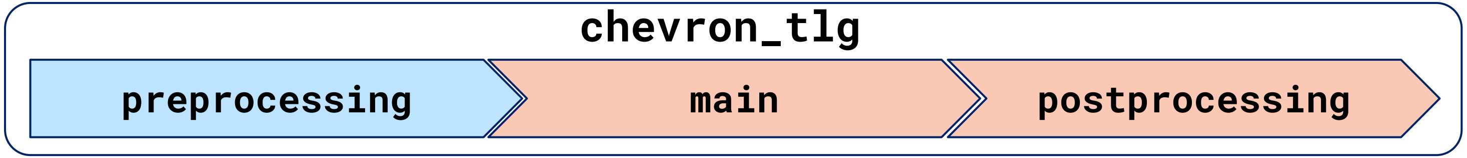
```
function(tlg, …) {
   tlg <- edit(tlg)
   return(tlg)
}
```

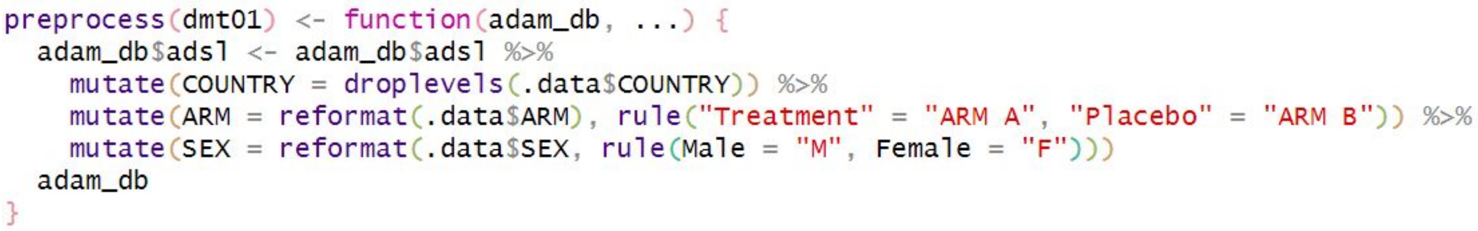
Expect: TLG
Return: processed TLG

E.g. Remove rows with zero counts

Extending {chevron}

Editing a chevron_tlg object



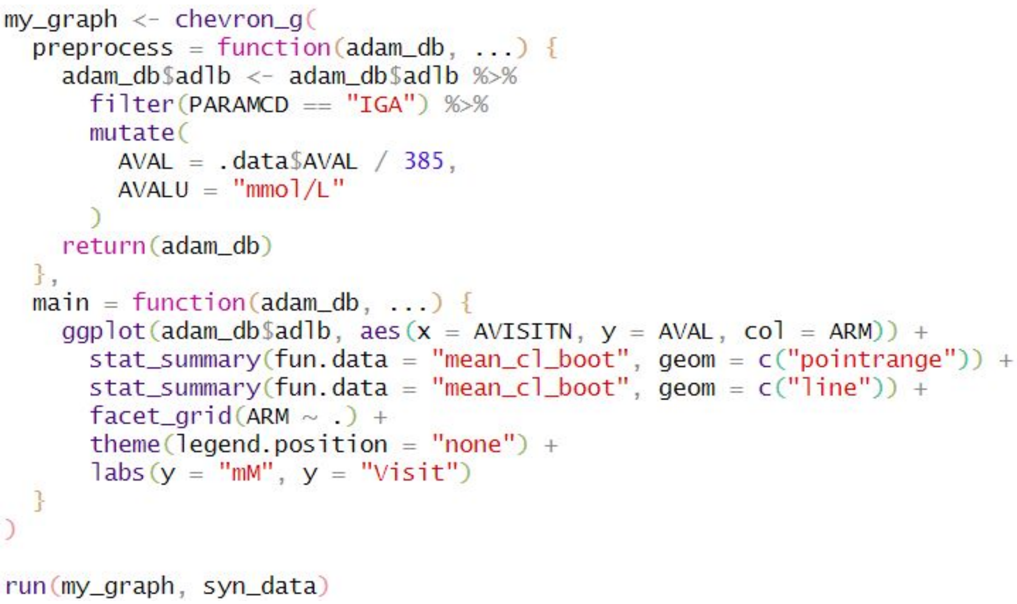


```
preprocess(dmt01) <- function(adam_db, ...) {
  adam_db$adsl <- adam_db$adsl %>%
    mutate(COUNTRY = droplevels(.data$COUNTRY)) %>%
    mutate(ARM = reformat(.data$ARM), rule("Treatment" = "ARM A", "Placebo" = "ARM B")) %>%
    mutate(SEX = reformat(.data$SEX, rule(Male = "M", Female = "F")))
  adam_db
}
```

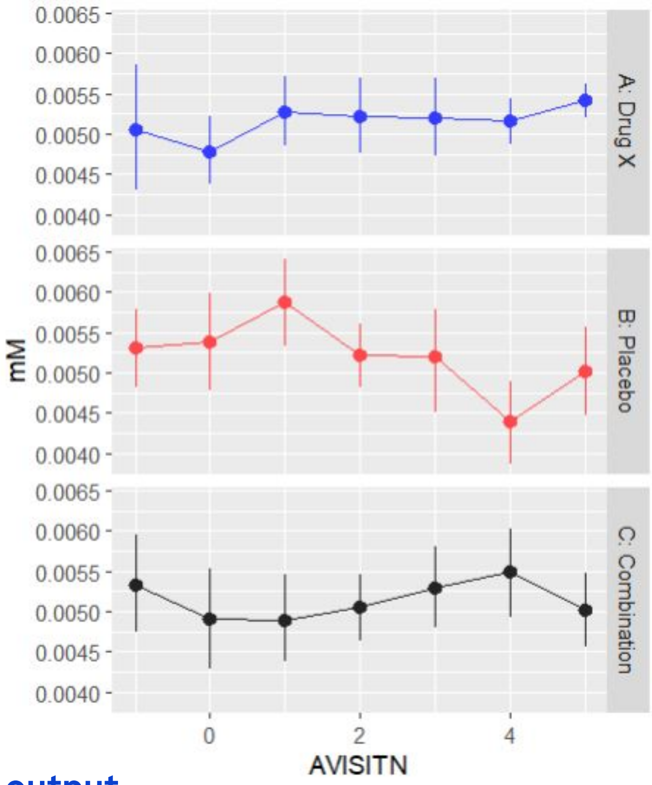
- The new function must respect some (reasonable) conventions (see doc).
- The other slots (main and postprocessing) can also be edited.
- Data manipulation can also be conducted before passing data to chevron.

Extending {chevron}

Creating a new chevron object



```
my_graph <- chevron_g(
  preprocess = function(adam_db, ...) {
    adam_db$adlb <- adam_db$adlb %>%
      filter(PARAMCD == "IGA") %>%
      mutate(
        AVAL = .data$AVAL / 385,
        AVALU = "mmol/L"
      )
    return(adam_db)
  },
  main = function(adam_db, ...) {
    ggplot(adam_db$adlb, aes(x = AVISITN, y = AVAL, col = ARM)) +
      stat_summary(fun.data = "mean_cl_boot", geom = c("pointrange")) +
      stat_summary(fun.data = "mean_cl_boot", geom = c("line")) +
      facet_grid(ARM ~ .) +
      theme(legend.position = "none") +
      labs(y = "mM", y = "Visit")
  }
)

run(my_graph, syn_data)
```



- **Define some functions.**
- **Embed them into a chevron object according to the desired output.**
  - **`chevron_t`** (table), **`chevron_g`** (graph), **`chevron_l`** (listing)

{chevron} adoption in Roche

- Already adopted in Roche!
  - Part of the Next-Gen toolkit to create standard outputs
  - Use by multiple trials, including early and late phase studies
- Testimony from the user
  - Most oncology efficacy and safety analysis are covered
  - Clear documentation to enable really quick generation of output
  - Easy to use and customize, reduce much effort

Using {chevron} today

Where to start

- **Install {chevron} from CRAN**

  ```
  install.packages(“chevron”)
  ```

- **Access the documentation**

  https://insightsengineering.github.io/chevron/latest-tag/

- **Explore the available templates**

  https://insightsengineering.github.io/chevron/latest-tag/articles/chevron_catalog.html

- **Collaborate with us**

  https://github.com/insightsengineering/chevron

Already covered standard outputs in {chevron}

AE Overview

AE frequency by SOC

Demographics

AE frequency by SOC and PT

Kaplan-Meier Plot

Time-to-Event and Survival Analysis

Cox-PH Analysis

Responder Analysis

Summary of Value and Change from Baseline

Patient Dispositions

Find more here!

Acknowledgement

- Thanks to Roche NEST team to make this happen!
- Thanks to Pawel and Joe for the excellent collaboration!
- Special thanks to Benoit, Chenkai, Lena, Tian, Tim, Xiaojin and Xiaoli for all the hard work in project chevron!

Doing now what patients need next