

Create TLGs and log files by sassy

Perphy ZHAO



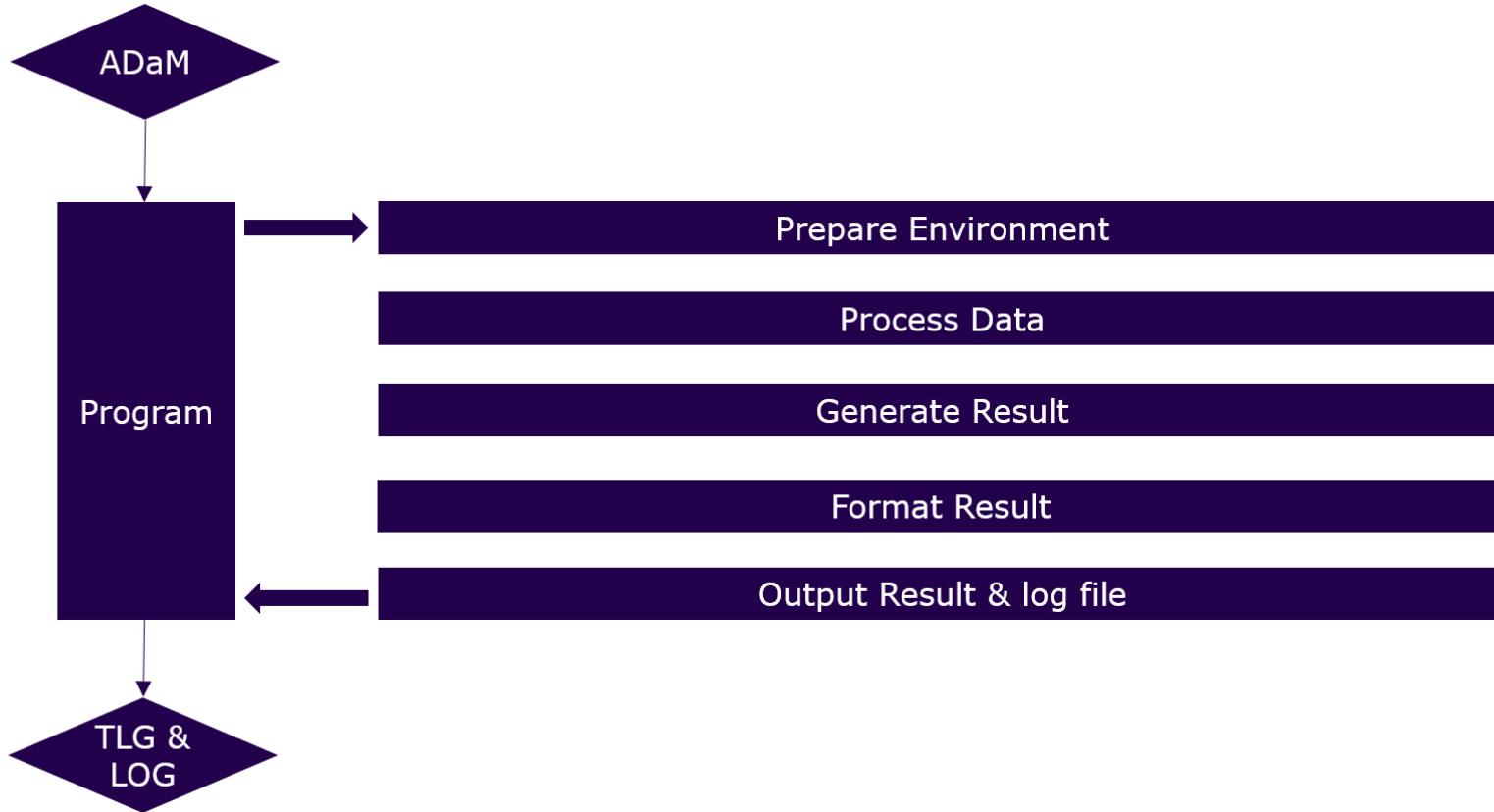
Agenda

- Introduction of sassy
- TLGs Programming & Log Output
- Functions Deep Dive
- Summary
- Q&A

Introduction of sassy

sanofi

Process Flow



Introduction of sassy

sassy package makes R easier, especially pure SAS programmers, to create TLGs and log files. This package is a meta-package brings several SAS concept to R, and the programming grammar is highly similar with SAS.

- libr: defines libnames, generate data dictionaries, and simulate data steps.
- fmtr: provides functions format data and creating format catalogs.
- procs: functions simulate SAS procedures and includes simulations of FREQ, MEANS, TRANSPOSE, SORT and PRINT procedures.
- reporter: report with easy layout capabilities and the ability to write reports in RTF, DOCX, TXT and HTML file formats.
- logr: produces a traceable log files.
- common: utility functions across the sassy family packages, and useful in their own right.

TLGs Programming & Log Output

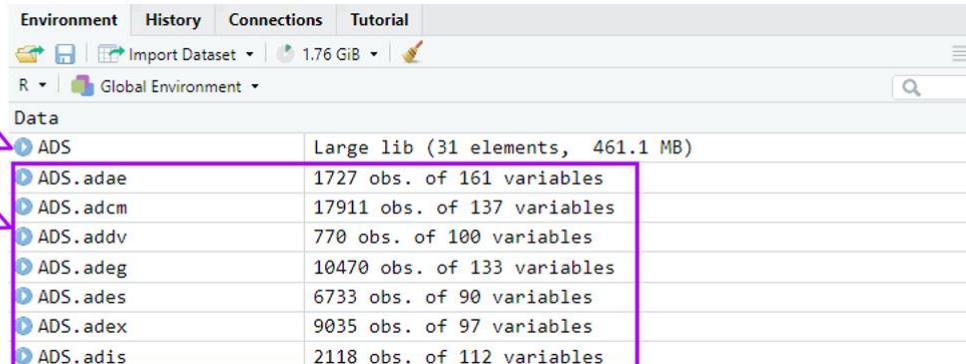
sanofi

Table Programming

```
# Prepare Environment -----  
  
library(tidyverse)  
library(haven)  
library(sassy)  
  
ADD <- "STUDY_TRIPLET/CSR/ADS/DATA"           # datasets path  
OUTOUT <- "STUDY_TRIPLET/CSR/REPORT/OUTPUT"    # output path  
LOG <- "STUDY_TRIPLET/CSR/REPORT/LOG"          # log path  
  
libname(ADS, ADD, "sas7bdat")                  # create a libname ADS  
lib_load(ADS)                                  # load data into workspace
```

```
# library 'ADS': 31 items  
- attributes: sas7bdat not loaded           /CSR/ADS/DATA  
- path:  
- items:  
  
1      Name Extension  Rows Cols   Size  LastModified  
2      adae sas7bdat  1727 161   2.4 Mb 2022-01-17 03:24:37  
3      adcm sas7bdat  17911 137  18.8 Mb 2022-01-17 03:25:17  
4      addv sas7bdat  770 100  816.9 Kb 2022-01-17 03:39:23  
5      adeg sas7bdat  10470 133  10.7 Mb 2022-01-17 03:28:15  
6      ades sas7bdat  6733 90   4.8 Mb 2022-01-17 03:27:49  
7      adex sas7bdat  9035 97   7.3 Mb 2022-01-17 03:27:09  
     adis sas7bdat  2118 112   2 Mb  2022-01-17 03:38:41
```

```
# library 'ADS': 31 items  
- attributes: sas7bdat loaded           /CSR/ADS/DATA  
- path:  
- items:  
  
1      Name Extension  Rows Cols   Size  LastModified  
2      adae sas7bdat  1727 161   2.4 Mb 2022-01-17 03:24:37  
3      adcm sas7bdat  17911 137  18.8 Mb 2022-01-17 03:25:17  
4      addv sas7bdat  770 100  816.9 Kb 2022-01-17 03:39:23  
5      adeg sas7bdat  10470 133  10.7 Mb 2022-01-17 03:28:15  
6      ades sas7bdat  6733 90   4.8 Mb 2022-01-17 03:27:49  
7      adex sas7bdat  9035 97   7.3 Mb 2022-01-17 03:27:09  
     adis sas7bdat  2118 112   2 Mb  2022-01-17 03:38:41
```



The screenshot shows the RStudio interface with the Global Environment tab selected. A table lists the datasets in the ADS library, each with its name, extension, number of observations (Rows), number of variables (Cols), size, and last modified date. The first dataset, ADS.adae, is highlighted with a red border.

	Name	Extension	Rows	Cols	Size	LastModified
1	ADS.adae	sas7bdat	1727	161	2.4 Mb	2022-01-17 03:24:37
2	ADS.adcm	sas7bdat	17911	137	18.8 Mb	2022-01-17 03:25:17
3	ADS.addv	sas7bdat	770	100	816.9 Kb	2022-01-17 03:39:23
4	ADS.adeg	sas7bdat	10470	133	10.7 Mb	2022-01-17 03:28:15
5	ADS.ades	sas7bdat	6733	90	4.8 Mb	2022-01-17 03:27:49
6	ADS.adex	sas7bdat	9035	97	7.3 Mb	2022-01-17 03:27:09
7	ADS.adis	sas7bdat	2118	112	2 Mb	2022-01-17 03:38:41

Table Programming

```
# Prepare data
ads1 <- datastep(ADS.ads1,
  keep = c('USUBJID', 'SEX', 'AGE', 'RACE', 'SAFFL',
    find.names(ADS.ads1, pattern = 'TRTGRP*'),
    'AGEGRP', 'AGEGRPN'),
  where = expression(SAFFL == 'Y'),
  {
    if (AGE > 10)
    {
      AGEGRP = '> 10'
      AGEGRPN = 1L
    }
    else
    {
      AGEGRP = '<= 10'
      AGEGRPN = 2L
    }
  }
# any operation can be done in {} like sas
})
```

keep option is like retain statement in SAS, can retain the variables' order

	USUBJID	Sex	Age	Race	SAFFL	TREATMENT	TRTGRP1	TRTGRP1N	AGEGRP	AGEGRPN
1	00001	M	11	WHITE	Y	Treatment			> 10	1
2	00002	M	13	WHITE	Y	Treatment			> 10	1
3	00003	M	12	WHITE	Y	Placebo			> 10	1
4	00004	M	9	WHITE	Y	Placebo			<= 10	2
5	00005	F	8	WHITE	Y	Placebo			<= 10	2
6	00006	F	15	WHITE	Y	Treatment			> 10	1
7	00007	M	8	WHITE	Y	Treatment			<= 10	2
8	00008	F	13	WHITE	Y	Treatment			> 10	1
9	00009	M	14	OTHER	Y	Placebo			> 10	1

Table Programming

```
# Age Summary  
age_sum <- proc_means(adsl, var = AGE,  
                      stats = v(n, mean, std, median, q1, q3, min, max),  
                      by = TRTGRP1,  
                      options = v(notype, nofreq))
```

```
# Format the stats  
fc <- fcat(MEAN = '%.1f', STD = ' (%.2f)',  
            Q1 = '%.1f', Q3 = '%.1f',  
            MIN = '%d', MAX = '%d')
```

```
age_fmt <- datastep(age_sum,  
                     format = fc,  
                     drop = find.names(age_sum, start = 4),  
                     {  
                     `^Mean (SD)` <- fapply2(MEAN, STD)  
                     Median <- MEDIAN  
                     `^Q1 : Q3` <- fapply2(Q1, Q3, sep = " : ")  
                     `^Min : Max` <- fapply2(MIN, MAX, sep = " : ")  
                     })
```

```
age_tran <- proc_transpose(age_fmt,  
                           var = names(age_fmt),  
                           copy = VAR, id = BY,  
                           name = LABEL)
```

Packages Help Viewer Presentation

TRTGRP1=Placebo									
Variable	N	Mean	Std Dev	Median	Lower Quartile	Upper Quartile	Minimum	Maximum	
AGE	31	12.7741935	1.7834363	12.0000000	12.0000000	15.0000000	9.0000000	16.0000000	

TRTGRP1=Treatment									
Variable	N	Mean	Std Dev	Median	Lower Quartile	Upper Quartile	Minimum	Maximum	
AGE	69	12.7826087	2.0641874	13.0000000	11.0000000	14.0000000	8.0000000	17.0000000	

age_sum x										
BY	VAR	N	MEAN	STD	MEDIAN	Q1	Q3	MIN	MAX	
1	Placebo	AGE	31	12.77419	1.783436	12	12	15	9	16
2	Treatment	AGE	69	12.78261	2.064187	13	11	14	8	17

age_fmt x										
BY	VAR	N	Mean (SD)	Median	Q1 : Q3	Min : Max				
1	Placebo	AGE	31	12.8 (1.78)	12	12.0 : 15.0	9 : 16			
2	Treatment	AGE	69	12.8 (2.06)	13	11.0 : 14.0	8 : 17			

age_tran x				
VAR	LABEL	Placebo	Treatment	
1	AGE	N	31	69
2	AGE	Mean (SD)	12.8 (1.78)	12.8 (2.06)
3	AGE	Median	12	13
4	AGE	Q1 : Q3	12.0 : 15.0	11.0 : 14.0
5	AGE	Min : Max	9 : 16	8 : 17

Table Programming

```
# Sex frequency  
sex_freq <- proc_freq(adsl, tables = SEX,  
                      by = TRTGRP1,  
                      options = 'nonobs')  
  
# Format the stats  
sex_fmt <- value(condition(x == 'F', 'Female', order = 1L),  
                  condition(x == 'M', 'Male', order = 2L))  
  
fc <- fcat(SEX = sex_fmt,  
            CNT = '%2d',  
            PCT = '(%5.2f%%)')  
  
sex_fmt <- datastep(sex_freq,  
                     format = fc,  
                     drop = v(CNT,PCT),  
                     rename = list(CAT = 'LABEL'),  
                     {  
                       CNTPCT = fapply2(CNT, PCT)  
                       CAT <- fapply(CAT, fc$SEX)  
                     })  
  
sex_tran <- proc_transpose(sex_fmt,  
                           var = CNTPCT,  
                           copy = VAR, id = BY,  
                           by = LABEL)
```

The screenshot shows the SAS interface with three data tables and two SAS datasets.

Table of TRTGRP1=Placebo, Sex

SEX	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	12	38.71	12	38.71
M	19	61.29	31	100.00

Table of TRTGRP1=Treatment, Sex

SEX	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	19	27.54	19	27.54
M	50	72.46	69	100.00

sex_freq

BY	VAR	CAT	CNT	PCT	
1	Placebo	SEX	F	12	38.70968
2	Placebo	SEX	M	19	61.29032
3	Treatment	SEX	F	19	27.53623
4	Treatment	SEX	M	50	72.46377

sex_fmt

BY	VAR	LABEL	CNTPCT
1	Placebo	SEX	Female 12 (38.71%)
2	Placebo	SEX	Male 19 (61.29%)
3	Treatment	SEX	Female 19 (27.54%)
4	Treatment	SEX	Male 50 (72.46%)

sex_tran

VAR	LABEL	NAME	Placebo	Treatment
1	SEX	Female	CNTPCT	12 (38.71%)
2	SEX	Male	CNTPCT	19 (61.29%)

Table Programming

```
# Combine all block  
dem_demo_s_t <- datastep(sex_tran,  
                           set = list(race_tran, age_tran, age_tran1),  
                           drop = v(NAME),  
                           {})
```

VAR	LABEL	NAME	Placebo	Treatment
1	SEX	Female	CNTPCT	12 (38.71%) 19 (27.54%)
2	SEX	Male	CNTPCT	19 (61.29%) 50 (72.46%)

VAR	LABEL	NAME	Placebo	Treatment
1	RACE	White	CNTPCT	30 (96.77%) 65 (94.20%)
2	RACE	Black or African American	CNTPCT	0 (0.00%) 2 (2.90%)
3	RACE	Other	CNTPCT	1 (3.23%) 2 (2.90%)

VAR	LABEL	Placebo	Treatment
1	AGE	N	31 69
2	AGE	Mean (SD)	12.8 (1.78) 12.8 (2.06)
3	AGE	Median	12 13
4	AGE	Q1 : Q3	12.0 : 15.0 11.0 : 14.0
5	AGE	Min : Max	9 : 16 8 : 17

VAR	LABEL	NAME	Placebo	Treatment
1	AGEGRP	<= 10	CNTPCT	2 (6.45%) 7 (10.14%)
2	AGEGRP	> 10	CNTPCT	29 (93.55%) 62 (89.86%)

VAR	LABEL	Placebo	Treatment
1	SEX	Female	12 (38.71%) 19 (27.54%)
2	SEX	Male	19 (61.29%) 50 (72.46%)
3	RACE	White	30 (96.77%) 65 (94.20%)
4	RACE	Black or African American	0 (0.00%) 2 (2.90%)
5	RACE	Other	1 (3.23%) 2 (2.90%)
6	AGE	N	31 69
7	AGE	Mean (SD)	12.8 (1.78) 12.8 (2.06)
8	AGE	Median	12 13
9	AGE	Q1 : Q3	12.0 : 15.0 11.0 : 14.0
10	AGE	Min : Max	9 : 16 8 : 17
11	AGEGRP	<= 10	2 (6.45%) 7 (10.14%)
12	AGEGRP	> 10	29 (93.55%) 62 (89.86%)

Table Programming

```
# Report
var_fmt <- c('AGE' = 'Age(years)', 'AGEGRP' = 'Age Group',
           'SEX' = 'Sex', 'RACE' = 'Race')

# Create Table
tbl <- create_table(dem_demo_s_t, first_row_blank = FALSE) %>%
  column_defaults(c('Placebo', 'Treatment'), align = 'center', width = 3.65) %>%
  stub(vars = c('VAR', 'LABEL'), 'Demographic Category', width = 4.5) %>%
  define(VAR, blank_after = TRUE, dedupe = TRUE, label = '',
         format = var_fmt, label_row = TRUE) %>%
  define(LABEL, indent = .25, label = '') %>%
  define(Placebo, label = 'Placebo', n = bigN['Placebo'],
         align = 'center') %>%
  define(Treatment, label = 'Treatment', n = bigN['Treatment'],
         align = 'center') %>%
  titles('Demographic data, data at baseline and medication details',
         'Demographics',
         'Patient characteristics at baseline - Safety population',
         bold = TRUE,
         align = 'left',
         borders = 'bottom') %>%
  footnotes('PGM=.../CSR/REPORT/PGM/dem_demo_s_t.R
            OUT=OUTPUT/dem_demo_mod_s_t_x.rtf',
            borders = 'top',
            blank_row = 'none')

rpt <- create_report(file.path(QCO, "dem_demo_s_t"),
                      output_type = "RTF",
                      font = "Arial") %>%
  page_header('Sponsor: Sanofi', 'Study: XXX') %>%
  set_margins(left = 1.1, right = 0.79, top = 1.3, bottom = 0.67) %>%
  add_content(tbl) %>%
  page_footer('Date Produced: {Sys.Date()}', right = 'Page [pg] of [tpp]')
res <- write_report(rpt)
```

VAR	LABEL	Placebo	Treatment
1 SEX	Female	12 (38.71%)	19 (27.54%)
2 SEX	Male	19 (61.29%)	50 (72.46%)
3 RACE	White	30 (96.77%)	65 (94.20%)
4 RACE	Black or African American	0 (0.00%)	2 (2.90%)
5 RACE	Other	1 (3.23%)	2 (2.90%)
6 AGE	N	31	69
7 AGE	Mean (SD)	12.8 (1.78)	12.8 (2.06)
8 AGE	Median	12	13
9 AGE	Q1 : Q3	12.0 : 15.0	11.0 : 14.0
10 AGE	Min : Max	9 : 16	8 : 17

Demographic Category	Placebo (N=31)	Treatment (N=69)
Sex		
Female	12 (38.71%)	19 (27.54%)
Male	19 (61.29%)	50 (72.46%)
Race		
White	30 (96.77%)	65 (94.20%)
Black or African American	0 (0.00%)	2 (2.90%)
Other	1 (3.23%)	2 (2.90%)
Age(years)		
N	31	69
Mean (SD)	12.8 (1.78)	12.8 (2.06)
Median	12	13
Q1 : Q3	12.0 : 15.0	11.0 : 14.0
Min : Max	9 : 16	8 : 17
Age Group		
<= 10	2 (6.45%)	7 (10.14%)
> 10	29 (93.55%)	62 (89.86%)

PGM=.../CSR/REPORT/PGM/dem_demo_s_t.R OUT=OUTPUT/dem_demo_mod_s_t_x.rtf

Listing Programming

```

# Prepare list data set
# Create list
tbl <- create_table(final1, first_row_blank = FALSE) %>%
  column_defaults(c('TRTDEMO', 'VISIT_', 'HGB', 'HCT', 'PLAT')) %>%
  spanning_header(from = 3, to = 5,
    label = "Parameter(Unit)\nvalue/Change from baseline{supsc('c')})" ) %>%
  define(TRTDEMO, [blank_after = TRUE], [dedupe = TRUE],
    align = 'left', [label_align = 'left', width = 2.6,
    label = "Arm/Patient\n(Age,Sex{supsc('a')},\nRace{supsc('b')},Weight(kg))")
  ) %>%
  define(VISIT_, align = 'left', width = 1.6, label_align = 'left',
    label = "Measurement date{supsc('c')}/Dose administration\non the same day") %>%
  define(HGB, align = 'centre', width = 1.6,
    label = 'Hemoglobin\n(G/L)') %>%
  define(HCT, align = 'centre', width = 1.6,
    label = 'Hematocrit') %>%
  define(PLAT, align = 'centre', width = 1.6,
    label = 'Plalets\n(GIGA/L)')
  total width is about 9 for the horizontal
rpt <- create_report(file.path(QCO, "lab_pcsa_rbc_s_1"),
  output_type = "RTF",
  font = "Times") %>%
page_header('Sponsor: Sanofi', 'Study: XXX') %>%
set_margins(left = 1.1, right = 0.79, top = 1.3, bottom = 0.67) %>%
add_content(tbl) %>%
page_footer('Date Produced: {Sys.Date()}', right = 'Page [pg] of [tpg]')
res <- write_report(rpt)

```

final1 x

Filter

TRTDEMO	VISIT_	HGB	HCT	PLAT
1 Placebo/0001(10,M,W,32)	2017-08-15/D-364/N	129/0	0.39/0	184/0
2 Placebo/0001(10,M,W,32)	2018-08-14/D1/Y	130/1	0.39/-0.004	178/-6
3 Placebo/0001(10,M,W,32)	2018-09-11/D29/Y	136/7	0.45/0.06	252/68
4 Placebo/0001(10,M,W,32)	2018-11-05/D84/Y	128/-1	0.45/0.061	223/39
5 Placebo/0001(10,M,W,32)	2019-01-30/D170/Y	133/4	0.47/0.084+	219/35

Sponsor: Sanofi
Clinical laboratory data
Listings
Red blood cells, platelets and coagulation - Listing of patients with abnormalities (PCSA) during the TEAE period - Safety population

Study: XXX

Parameter(Unit) value/Change from baseline*	Hemoglobin (G/L)	Hematocrit	Plalets (GIGA/L)
Arm/Patient (Age,Sex,Race),Weight(kg)) Placebo/0001(10,M,W,32)			
Measurement date/Dose administration on the same day			
2017-08-15/D-364/N	129/0	0.39/0	184/0
2018-08-14/D1/Y	130/1	0.39/-0.004	178/-6
2018-09-11/D29/Y	136/7	0.45/0.06	252/68
2018-11-05/D84/Y	128/-1	0.45/0.061	223/39
2019-01-30/D170/Y	133/4	0.47/0.084+	219/35
2019-04-24/D254/Y	128/-1	0.44/0.05	203/19
2019-08-14/D366/N	132/3	0.45/0.061	227/43
2019-11-11/D455/N	129/0/1	0.44/0.048	210/26
Treatment/0002(9,M,W,28)			
2017-09-28/D-364/N	151/0	0.45/0	308/0
2018-09-27/D1/Y	132/-19	0.39/-0.063	295/-13
2018-10-25/D29/Y	139/-12	0.44/-0.012	322/14
2019-01-02/D98/Y	132/-19	0.43/-0.026	326/18
2019-03-14/D169/Y	128/-23	0.42/-0.033	300/-8
2019-06-06/D253/Y	130/-21	0.42/-0.029	295/-13
2019-09-23/D362/N	134/-17	0.43/-0.025	365/57
2020-01-27/D488/N	133/-18	0.44/-0.014	362/54

*M=Male, F=Female
*W=Caucasian/White, B=Black, A=Asian/Oriental, I=American Indian or Alaska Native, N=Native Hawaiian or Other Pacific Islander, O=Other
*U=Unscheduled
*Relative day to the first dose of XXX
*L: below lower limit of normal (LLN); H: above upper limit of normal (ULN); - or +: values reaching the lower or upper PCSA limit
PGM=...CSR/REPORT/PGM/lab_pcsa_s_1.R OUT=OUTPUT/lab_pcsa_ful_s_1_x.rtf

Figure Programming

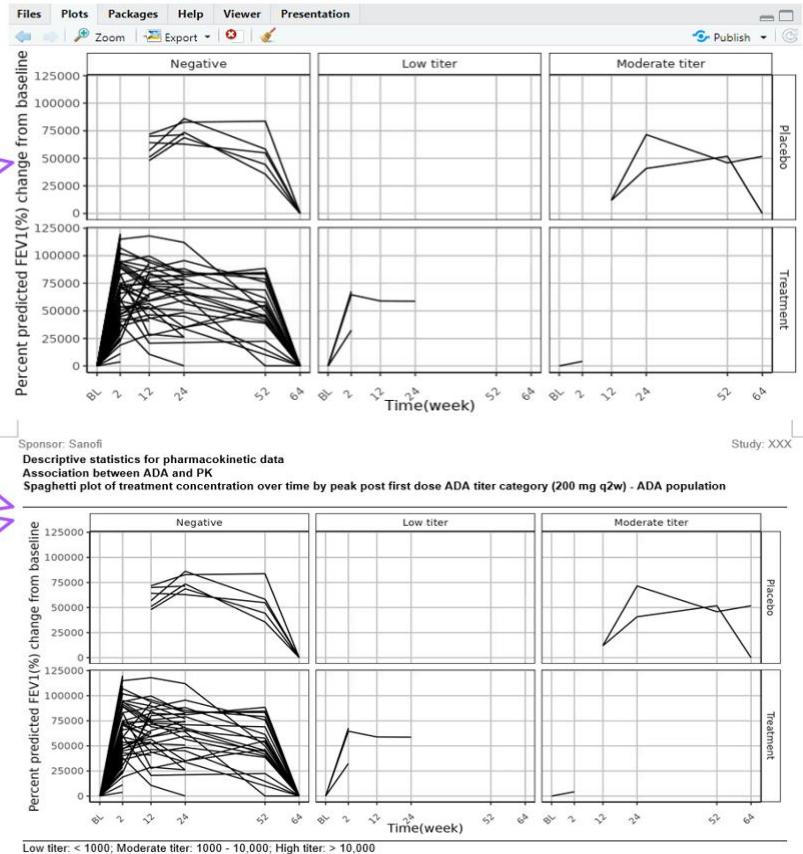
```
# Prepare data for plot
# can use tidyverse, sassy to prepare, or any other package

# Plot
# can use ggplot, survfit to plot, or any other jpg format
p_plot <- ggplot(final, aes(x = week, y = AVAL,
                             group = as.factor(USUBJID)))

# Create plot object and report
page <- create_plot(p_plot, 4.3, 9) %>%
  titles('Descriptive statistics for pharmacokinetic data',
        'Association between ADA and PK',
        'Spaghetti plot of treatment concentration over time by peak post
        first dose ADA titer category (200 mg q2w) - ADA population',
        bold = TRUE,
        align = 'left',
        borders = 'bottom') %>%
  footnotes('Low titer: < 1000; Moderate titer: 1000 - 10,000;
            High titer: > 10,000',
            'PGM=.../CSR/REPORT/PGM/pk_peak_ada_titer_a_g
            OUT=OUTPUT/pk_peak_ada_titer_a_g_x.rtf',
            borders = 'top',
            blank_row = 'none')

rpt <- create_report(file.path(QCO, 'pk_peak_ada_titer_a_g'),
                      output_type = 'RTF',
                      font = 'Arial') %>%
  set_margins(left = 1.1, right = 0.79, top = 1.3, bottom = 0.67) %>%
  page_header('Sponsor: Sanofi', 'Study: XXX') %>%
  add_content(page) %>%
  page_footer(paste0('Date Produced: ', fapply(Sys.time(), '%d%b%y %H:%M')),
              right = "Page [pg] of [tpg]")

write_report(rpt)
```



Log Output

```
# Set options for log
options("logr.autolog" = TRUE,
       "logr.on" = TRUE,
       "logr.notes" = TRUE,
       "procs.print" = TRUE,
       "tidylog.display" = list(log_print))

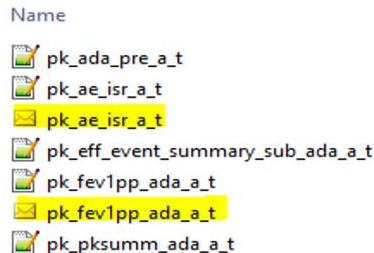
# Open log
lgpth <- log_open(file.path(QCL, "dem_demo_s_t.log"),
                   logdir = FALSE,
                   autolog = NULL,
                   compact = FALSE,
                   traceback = TRUE)

# Input your code
```

log for the demographic age summary block

```
# Close log
log_close()
```

Note: there will be an outlook file in your log folder if your program has error(s)



```
[ dem_demo_s_t.log ]
```

```
1 =====
2 Log Path: /QC/LOG/dem_demo_s_t.log
3 Program Path: /HIS/C/test/pharma R2024.R
4 Working Directory: /CSR_R
5 User Name:
6 R Version: 4.2.1 (2022-06-23)
7 Machine: w-ride-prod-01.pharma.aventis.com x86_64
8 Operating System: Linux 3.10.0-1160.e17.x86_64 #1 SMP Tue Aug 18 14:50:17 EDT 2020
9 Base Packages: stats graphics grDevices utils datasets methods base Other Packages: tidylog_1.0.2
10 Log Start Time: 2024-02-17 12:10:29
11 =====
12 # A user-defined format: 2 conditions
13   Name Type Expression Label Order
14   1   x   U   x <= 10 <= 10   1
15   2   x   U   x > 10 > 10   2
16
17 NOTE: Log Print Time: 2024-02-17 12:10:34
18 NOTE: Elapsed Time: 5.464285138526517 secs
19
20 # A user-defined format: 2 conditions
21   Name Type Expression Label Order
22   1   x   U   x == "F" Female   1
23   2   x   U   x == "M" Male     2
24
25 NOTE: Log Print Time: 2024-02-17 12:10:36
26 NOTE: Elapsed Time: 1.67262363433838 secs
27
28 # A user-defined format: 3 conditions
29   Name Type Expression Label Order
30   1   x   U   x == "WHITE" White    1
31   2   x   U   x == "BLACK OR AFRICAN AMERICAN" Black or African American 2
32   3   x   U   x == "OTHER" Other    3
33
34 NOTE: Log Print Time: 2024-02-17 12:10:37
35 NOTE: Elapsed Time: 0.880235910415649 secs
36
37 Compile format catalog
38
39 NOTE: Log Print Time: 2024-02-17 12:10:38
40 NOTE: Elapsed Time: 1.1965569473053 secs
41
42 # A format catalog: 11 formats
43 - $AGEGRP: type U, 2 conditions
44 - $SEX: type U, 2 conditions
45 - $RACE: type U, 3 conditions
46 - $MEAN: type S, "%_1f"
47 - $STD: type S, "%._2f"
48 - $Q1: type S, "%._1f"
49 - $Q3: type S, "%._1f"
50 - $MIN: type S, "%d"
51 - $MAX: type S, "%d"
52 - $CNT: type S, "%2d"
53 - $PCI: type S, "(%.1f%%)"
54
55 NOTE: Log Print Time: 2024-02-17 12:10:39
56 NOTE: Elapsed Time: 1.06215500831604 secs
57
58 datastep: columns started with 11 and ended with 11
59
60 NOTE: Log Print Time: 2024-02-17 12:10:41
61 NOTE: Elapsed Time: 1.33687615394592 secs
62
63 # A tibble: 100 × 11
64   USUBJID SEX AGE RACE SAFFL TRTGRPL  TRIGRPLN AGEGRP AGEGRPN SEX_FMT RACE_FMT
65   <chr>   <chr> <dbl> <chr> <chr> <chr> <int> <chr> <chr>
66   1 00001   M    17  WHI   F   17  WHI   1  Male  White
67   2 00002   M    17  WHI   F   17  WHI   2  Male  White
68   3 00003   M    17  WHI   F   17  WHI   3  Male  White
69   4 00004   M    17  WHI   F   17  WHI   4  Male  White
70   5 00005   M    17  WHI   F   17  WHI   5  Male  White
71   6 00006   M    17  WHI   F   17  WHI   6  Male  White
72   7 00007   M    17  WHI   F   17  WHI   7  Male  White
73   8 00008   M    17  WHI   F   17  WHI   8  Male  White
74   9 00009   M    17  WHI   F   17  WHI   9  Male  White
75  10 00010   M    17  WHI   F   17  WHI   10  Male  White
76  11 00011   M    17  WHI   F   17  WHI   11  Male  White
77  12 00012   M    17  WHI   F   17  WHI   12  Male  White
78  13 00013   M    17  WHI   F   17  WHI   13  Male  White
79  14 00014   M    17  WHI   F   17  WHI   14  Male  White
80  15 00015   M    17  WHI   F   17  WHI   15  Male  White
81  16 00016   M    17  WHI   F   17  WHI   16  Male  White
82  17 00017   M    17  WHI   F   17  WHI   17  Male  White
83  18 00018   M    17  WHI   F   17  WHI   18  Male  White
84  19 00019   M    17  WHI   F   17  WHI   19  Male  White
85  20 00020   M    17  WHI   F   17  WHI   20  Male  White
86  21 00021   M    17  WHI   F   17  WHI   21  Male  White
87  22 00022   M    17  WHI   F   17  WHI   22  Male  White
88  23 00023   M    17  WHI   F   17  WHI   23  Male  White
89  24 00024   M    17  WHI   F   17  WHI   24  Male  White
90  25 00025   M    17  WHI   F   17  WHI   25  Male  White
91  26 00026   M    17  WHI   F   17  WHI   26  Male  White
92  27 00027   M    17  WHI   F   17  WHI   27  Male  White
93  28 00028   M    17  WHI   F   17  WHI   28  Male  White
94  29 00029   M    17  WHI   F   17  WHI   29  Male  White
95  30 00030   M    17  WHI   F   17  WHI   30  Male  White
96  31 00031   M    17  WHI   F   17  WHI   31  Male  White
97  32 00032   M    17  WHI   F   17  WHI   32  Male  White
98  33 00033   M    17  WHI   F   17  WHI   33  Male  White
99  34 00034   M    17  WHI   F   17  WHI   34  Male  White
100 35 00035   M    17  WHI   F   17  WHI   35  Male  White
```

Functions Deep Dive

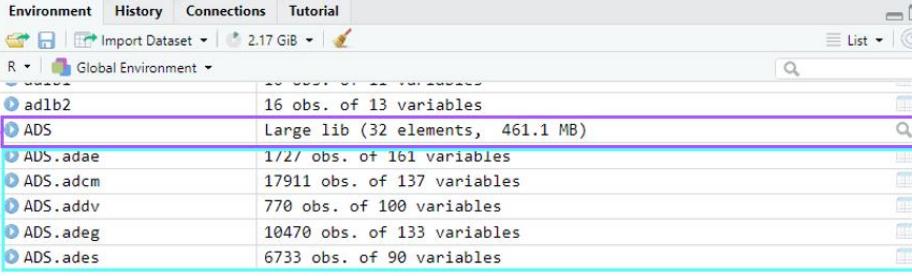
sanofi

Functions in libr

- `datastep`: Steps through data row-by-row, similar with SAS data step.
- `delete`: Removes records from a datastep.
- `dictionary`: Creates a data dictionary, like SAS contents procedure.
- `dsarry`: Creates a data step array.
- `dsattr`: Assigns datastep variable attributes.
- `libname`: Creates a data library, like libname statement in SAS.
- `lib_add`: Adds data to a data library.
- `lib_copy`: Copies a data library.
- `lib_delete`: Deletes a data library.
- `lib_export`: Exports a data library.
- `lib_load`: Loads a library into the workspace.
- `lib_remove`: Removes data from a data library.
- `lib_unload`: Unloads a library from the workspace.
- `lib_path`: Gets the path for a data library.
- `output`: Outputs records from a datastep.

Call functions in libr

```
# Create a data library ADS  
libname(ADS, MADD, "sas7bdat")  
  
# Load the library into the workspace  
lib_load(ADS)  
  
# Create a data dictionary, like SAS contents procedure  
content <- dictionary(ADS)  
  
# Add data final1 to the library  
lib_add(ADS, final1)  
  
# Remove data final1 to the library  
lib_remove(ADS, final1)
```



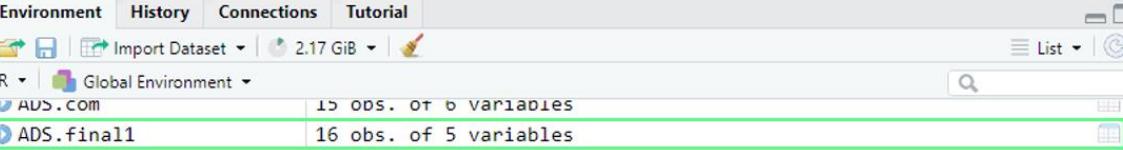
The screenshot shows the RStudio Global Environment pane. A purple arrow points from the first line of code, `libname(ADS, MADD, "sas7bdat")`, to the entry for 'ADS' in the list. Another purple arrow points from the second line of code, `lib_load(ADS)`, to the same entry. A pink arrow points from the third line of code, `content <- dictionary(ADS)`, to the 'content' table below.

Name	Column	Class	Label	Description	Format	Width	Justify	Rows	NAs	MaxChar
1 adae	STUDYID	character	Study Identifier	NA	\$15	NA	NA	1727	0	8
2 adae	USUBJID	character	Unique Subject Identifier	NA	\$25	NA	NA	1727	0	18
3 adae	SUBJID	character	Subject Identifier for the Study	NA	\$12	NA	NA	1727	0	12
4 adae	SITEID	character	Study Site Identifier	NA	NA	NA	NA	1727	0	7
5 adae	AGE	numeric	Age	NA	NA	NA	NA	1727	0	2
6 adae	AGEU	character	Age Units	NA	NA	NA	NA	1727	0	5
7 adae	SEX	character	Sex	NA	NA	NA	NA	1727	0	1
8 adae	BRTHDTC	character	Date/Time of Birth	NA	NA	NA	NA	1727	153	10
9 adae	BRTHDT	numeric	Date of Birth	NA	YYMMDD10	NA	NA	1727	153	5



The screenshot shows the RStudio Global Environment pane. A pink arrow points from the fourth line of code, `content <- dictionary(ADS)`, to this table. The table lists the variables in the 'content' of the ADS library.

Name	Column	Class	Label	Description	Format	Width	Justify	Rows	NAs	MaxChar
1 adae	STUDYID	character	Study Identifier	NA	\$15	NA	NA	1727	0	8
2 adae	USUBJID	character	Unique Subject Identifier	NA	\$25	NA	NA	1727	0	18
3 adae	SUBJID	character	Subject Identifier for the Study	NA	\$12	NA	NA	1727	0	12
4 adae	SITEID	character	Study Site Identifier	NA	NA	NA	NA	1727	0	7
5 adae	AGE	numeric	Age	NA	NA	NA	NA	1727	0	2
6 adae	AGEU	character	Age Units	NA	NA	NA	NA	1727	0	5
7 adae	SEX	character	Sex	NA	NA	NA	NA	1727	0	1
8 adae	BRTHDTC	character	Date/Time of Birth	NA	NA	NA	NA	1727	153	10
9 adae	BRTHDT	numeric	Date of Birth	NA	YYMMDD10	NA	NA	1727	153	5



The screenshot shows the RStudio Global Environment pane. A green arrow points from the fifth line of code, `lib_add(ADS, final1)`, to this table. The table lists the datasets in the ADS library.

Name	Column	Class	Label	Description	Format	Width	Justify	Rows	NAs	MaxChar
1 adae	STUDYID	character	Study Identifier	NA	\$15	NA	NA	1727	0	8
2 adae	USUBJID	character	Unique Subject Identifier	NA	\$25	NA	NA	1727	0	18
3 adae	SUBJID	character	Subject Identifier for the Study	NA	\$12	NA	NA	1727	0	12
4 adae	SITEID	character	Study Site Identifier	NA	NA	NA	NA	1727	0	7
5 adae	AGE	numeric	Age	NA	NA	NA	NA	1727	0	2
6 adae	AGEU	character	Age Units	NA	NA	NA	NA	1727	0	5
7 adae	SEX	character	Sex	NA	NA	NA	NA	1727	0	1
8 adae	BRTHDTC	character	Date/Time of Birth	NA	NA	NA	NA	1727	153	10
9 adae	BRTHDT	numeric	Date of Birth	NA	YYMMDD10	NA	NA	1727	153	5

datastep() function

- data: The data to step through.
- steps: The operations to perform on the data.
- keep: A vector of quoted variables names to keep in the output dataset, like keep statement in SAS.
- drop: A vector of quoted variables names to drop in the output dataset, like keep statement in SAS.
- rename: A named vector of quoted variables to rename, like rename statement in SAS.
- by: A vector of quoted variables to use for by-group processing, which will active the first. and last., automatic variables.
- calculate: Steps to set up calculated variables.
- retain: A list of variable names and initial values to retain, like retain statement in SAS.
- attrib: A named list of attributes, like attr statement in SAS.
- arrays: A named list of dsarry objects, like array statement in SAS.
- sort_check: Checks to see if the input data is sorted according to the by variable parameter.
- format: A named list of formats to assign to the output data frame, like format statement in SAS.
- label: A named list of labels to assign to the output data frame, like label statement in SAS.
- where: An expression to filter the output dataset.
- set: A dataset or list of datasets to append to the input data frame, like set statement in SAS.
- merge: A dataset or list of datasets to merge to the input data frame, like merge statement in SAS.
- merge_by: Be used to identify the variable(s) to merge by if merge the merge parameter us used.
- merge_in: A vector of column names to be used to hold the merge flags, like in option in SAS.
- log: Whether or not to log the datastep.

datastep()-Example

```
# Set two data sets, like set statement in SAS  
adsl_set <- datastep(data = adsl_F,  
                      set = adsl_M,  
                      {  
                        AGE1 = AGE + 2  
                      })
```

	USUBJID	SEX	AGE	RACE	SAFFL	TRTGRP1	TRTGRP1N
1	00016	F	13	WHITE	Y	Treatment	2

	USUBJID	SEX	AGE	RACE	SAFFL	TRTGRP1	TRTGRP1N
1	00014	M	13	WHITE	Y	Treatment	2
2	00047	M	13	WHITE	Y	Placebo	1
3	00065	M	13	WHITE	Y	Placebo	1
4	00089	M	13	WHITE	Y	Treatment	2

	USUBJID	SEX	AGE	RACE	SAFFL	TRTGRP1	TRTGRP1N	AGE1
1	00016	F	13	WHITE	Y	Treatment	2	15
2	00014	M	13	WHITE	Y	Treatment	2	15
3	00047	M	13	WHITE	Y	Placebo	1	15
4	00065	M	13	WHITE	Y	Placebo	1	15
5	00089	M	13	WHITE	Y	Treatment	2	15

datastep()-Example

```
# Inner join  
adsl_merge_inner <- datastep(data = adsl_arm,  
                               merge = adsl_age,  
                               merge_by = 'USUBJID',  
                               merge_in = c('ina', 'inb'),  
                               where = expression(ina & inb),  
                               {})  
  
# Left join  
adsl_merge_left <- datastep(data = adsl_arm,  
                               merge = adsl_age,  
                               merge_by = 'USUBJID',  
                               merge_in = c('ina', 'inb'),  
                               where = expression(ina),  
                               {})  
  
# Right join  
adsl_merge_right <- datastep(data = adsl_arm,  
                               merge = adsl_age,  
                               merge_by = 'USUBJID',  
                               merge_in = c('ina', 'inb'),  
                               where = expression(inb),  
                               {})  
  
# Full join  
adsl_merge_full <- datastep(data = adsl_arm,  
                               merge = adsl_age,  
                               merge_by = 'USUBJID',  
                               merge_in = c('ina', 'inb'),  
                               where = expression(ina | inb),  
                               {})
```

adsl_arm	
▲	USUBJID
1	00016
2	00014
3	00047
4	00065
5	00089

adsl_age	
▲	USUBJID
1	00016
2	00047

adsl_merge_inner				
		TRTGRP1	AGE	ina
1	00016	Treatment	13	1
2	00047	Placebo	13	1

adsl_merge_left				
		TRTGRP1	AGE	ina
1	00016	Treatment	13	1
2	00047	Placebo	13	1
3	00014	Treatment	NA	1
4	00065	Placebo	NA	1
5	00089	Treatment	NA	1

datastep()-Example

```
# Sort data set with proc_sort function in procs packages  
adsl_sort <- proc_sort(data = adsl_set,  
                        by = v(TRTGRP1N, TRTGRP1),  
                        keep = v(USUBJID, TRTGRP1N, TRTGRP1, AAGE))
```

```
# Retain & calculate options  
adsl_retain <- datastep(data = adsl_sort,  
                           by = c('TRTGRP1N', 'TRTGRP1'),  
                           retain = list(obs_num = 0, obs_num1 = 0),  
                           calculate = { mean_aage = mean(AAGE) })
```

```
    {  
        if (first.)  
            obs_num = 1  
        else  
            obs_num = obs_num+1  
    }
```

```
    {  
        if (first.)  
            obs_num1 = 0  
        obs_num1 = obs_num1+1  
    } )
```

Note: calculate option does not active by option

adsl_sort			
USUBJID	TRTGRP1N	TRTGRP1	AAGE
1 00047		1 Placebo	16
2 00065		1 Placebo	17
3 00016		2 Treatment	14
4 00014		2 Treatment	15
5 00089		2 Treatment	18

adsl_retain						
USUBJID	TRTGRP1N	TRTGRP1	AAGE	mean_aage	obs_num1	obs_num
1 00047		1 Placebo	16	16	1	1
2 00065		1 Placebo	17	16	2	2
3 00016		2 Treatment	14	16	1	1
4 00014		2 Treatment	15	16	2	2
5 00089		2 Treatment	18	16	3	3

Deep Dive in procs

- proc_freq: Generates frequency statistics.
- proc_means: Calculates summary statistics.
- proc_print: Prints a dataset.
- proc_sort: Sorts a dataset.
- proc_transpose: Transposes a dataset.
- proc_ttest: Calculates T-test statistics.

proc_freq() in procs

- data: The input dataframe to perform frequency calculations on.
- tables: The variables to perform frequency counts on.
- output: Whether or not to return datasets from function, valid values are ‘out’, ‘none’ and ‘report’, default is ‘out’, also accepts data shaping keywords ‘long’, ‘stacked’, ‘wide’.
- by: An optional by group, accepts a vector of one ore more variable names.
- weight: An optional weight parameter, which is passed as a variable name to use for the weight.
- options: The options desired for the function, which are passed to the parameter as a vector of quoted strings, can also use the v() finction to pass unquoted strings. The following options are available: ‘chiq’, ‘crosstab’, ‘fisher’, ‘list’, ‘missing’, ‘nlevels’, ‘ncol’, ‘nocum’, ‘nofreq’, ‘nopercent’, ‘noprint’, ‘noobs’, ‘norow’, ‘nosparse’, ‘notable’, ‘outcum’.

proc_freq()-Example

```
bigN_long<- proc_freq(adsl, tables = TRTGRP1,  
                      output = long,  
                      options = v(nopercent, nonobs))
```

VAR	STAT	Placebo	Treatment	
1	TRTGRP1	CNT	31	69

```
bigN_wide<- proc_freq(adsl, tables = TRTGRP1,  
                      output = wide,  
                      options = v(nopercent, nonobs))
```

VAR	CAT	CNT	
1	TRTGRP1	Placebo	31
2	TRTGRP1	Treatment	69

```
bigN_stack<- proc_freq(adsl, tables = TRTGRP1,  
                      output = stacked,  
                      options = v(nopercent, nonobs))
```

VAR	CAT	STAT	VALUES	
1	TRTGRP1	Placebo	CNT	31
2	TRTGRP1	Treatment	CNT	69

proc_means() in procs

- data: The input dataframe to perform summary statistics.
- var: The variable(s) to perform summary statistics for.
- output: Same as proc_freq.
- by: Same as proc_freq.
- class: The class parameter is similar to the by parameter, but the output is different. By groups will create completely separate tables, while class groups will be continued in the same table.
- stats: A vector of summary statistics keywords. Valid keywords are: "css", "clm", "cv", "kurt", "kurtosis", "lclm", "mean", "median", "mode", "min", "max", "n", "nmiss", "nobs", "p1", "p5", "p10", "p20", "p25", "p30", "p40", "p50", "p60", "p70", "p75", "p80", "p90", "p95", "p99", "q1", "q3", "qrange", "range", "skew", "skewness", "std", "stddev", "stderr", "sum", "uclm", "uss", and "vari". For hypothesis testing, the function supports "t", "prt", "probt", and "df". Default statistics are: "n", "mean", "std", "min", and "max".
- options: A vector of optional keywords. Valid values are: "alpha =", "completetypes", "maxdec =", "noprint", "notype", "nofreq", "nonobs", "nway". The "notype", "nofreq", and "nonobs" keywords will turn off columns on the output datasets. The "alpha = " option will set the alpha value for confidence limit statistics. The default is 95% (alpha = 0.05). The "maxdec = " option sets the maximum number of decimal places displayed on report output. The "nway" option returns only the highest type values.

proc_sort() in procs

- data: The input data to sort.
- by: A vector of variables to sort by.
- keep: A vector of variables on the output data to keep. All other variables will be dropped.
- order: The sort order of the variables on the by parameter. Valid values are 'ascending' or 'descending'. These values may also be abbreviated to 'asc', 'desc', 'a', or 'd'.
- options: Any options desired for the sort. Available options are 'dupkey' and 'nodupkey'. The 'nodupkey' option removes duplicate rows from the sorted dataset. The 'dupkey' option removes unique rows from the sorted dataset.
- as.character: If TRUE, will cast any factors in the 'by' parameter to character. Default is FALSE. This parameter is included because it is common to use factors for sorting in R, but you may not want to keep the variable as a factor. This parameter therefore allows you to use the factor for the sort, but then convert back to a character once the sort is complete.

proc_transpose() in procs

- data: The input data to transpose.
- by: An optional by group. Parameter accepts a vector of one or more quoted variable names. If the by group is requested, the data will be subset by that variable and the transpose function will transpose each group and stack them together in a single table.
- var: The variable(s) to transpose. Parameter accepts a vector of variable names. By default, all numeric variables will be transposed.
- id: The variable or variables to use for the transposed column names.
- idlabel: The variable to use for the transposed column labels.
- copy: A vector of variables to retain in the output data without transposition. Values will be truncated or recycled to fit the number of output rows.
- name: Specifies the name of the variable to be used for the var values.
- namelabel : The label to use for the name variable.
- prefix: Contains a prefix to be used in the construction of column names.
- delimiter: Specifies a delimiter to be used in the construction of column names.
- suffix: Contains a suffix to be used in the construction of column names.
- where: An expression to filter the rows after the transform is complete. Use the expression function to define the where clause.
- options: Optional keywords that affect the transpose. Default is NULL. Available option is "noname" which drops the name column from the output dataset.
- log: Whether or not to log the procedure. Default is TRUE. This parameter is used internally

proc_transpose()-Example

```
race_tran <- proc_transpose(race_fmt,
                             var = CNTPCT,
                             copy = VAR, id = BY,
                             by = v(CATN,CAT)) %>%
  proc_sort(by=v(CATN,CAT))
```

```
race_tran1 <- proc_transpose(race_fmt,
                             var = CNTPCT,
                             # copy = VAR,
                             id = BY,
                             by = v(VAR, CATN, CAT)) %>%
  proc_sort(by=v(CATN,CAT))
```

Variables in copy parameter can be passed to parameter

	BY	VAR	CAT RACE	CNTPCT	CATN
1	Placebo	RACE	Black or African American	0 (0.00%)	2
2	Placebo	RACE	Other	1 (3.23%)	3
3	Placebo	RACE	White	30 (96.77%)	1
4	Treatment	RACE	Black or African American	2 (2.90%)	2
5	Treatment	RACE	Other	2 (2.90%)	3
6	Treatment	RACE	White	65 (94.20%)	1

	VAR	CAT	CATN	NAME	Placebo	Treatment
1	RACE	White	1	CNTPCT	30 (96.77%)	65 (94.20%)
2	RACE	Black or African American	2	CNTPCT	0 (0.00%)	2 (2.90%)
3	RACE	Other	3	CNTPCT	1 (3.23%)	2 (2.90%)

	VAR	CATN	CAT	NAME	Placebo	Treatment
1	RACE	1	White	CNTPCT	30 (96.77%)	65 (94.20%)
2	RACE	2	Black or African American	CNTPCT	0 (0.00%)	2 (2.90%)
3	RACE	3	Other	CNTPCT	1 (3.23%)	2 (2.90%)

Deep Dive in logr

- log_close: Closes the log.
- log_code: A TRUE or FALSE value to indicate success or failure of the function.
- log_open: Initializes the log file.
- log_path: Gets the path to the currently opened log, which takes no parameters.
- log_print: Prints an object to the currently opened log.
- log_status: gets the status of the log. Possible status values are 'on', 'off', 'open', or 'closed', and which takes no parameters.

Deep Dive in fmtr

- condition: Creates a condition for a user-defined format, which used in conjunction with the value function.
- descriptions: Gets or sets descriptions for data frame columns
- fapply: Applies formatting to a vector.
- fapply2: Applies formatting to two different vectors, and combines them into a single vector.
- fattr: Assigns formatting attributes to a vector.
- fact: Creates a format catalog, which is a collection of formats.
- fdata: Applies formatting attributes to the entire data frame.
- fmt_cnt_pct, fmt_mean_sd, fmt_median, fmt_n, fmt_quantile_range, fmt_range: A family functions to calculate and format a count and percent, mean and standard deviation, median, numeric count, quantile range, numeric range.
- formats: Gets or sets formats for a data frame.
- value: Creates a user-defined format.
- width: Gets or sets column widths for a data frame.

fmt family-Example

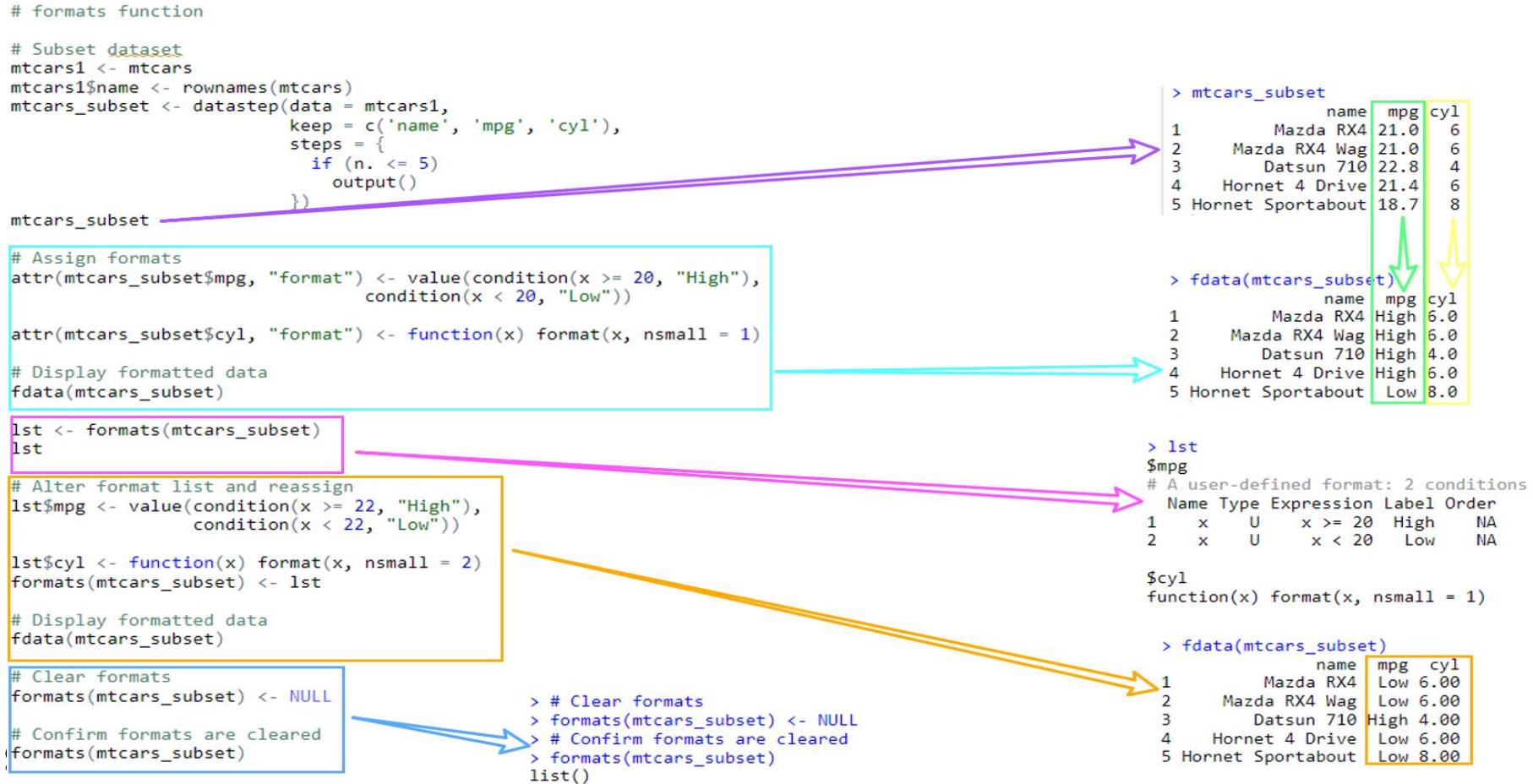
```
# Format family function
Fmt_family <- proc_sort(data = adsl,
                         by = v(TRTGRP1N, TRTGRP1)) %>%
  datastep(by = v(TRTGRP1N, TRTGRP1),
           calculate = { age_mean_sd = fmt_mean_sd(AGE)
                         age_n = fmt_n(AGE)
                         age_median =fmt_median(AGE)
                         age_quantile =fmt_quantile_range(AGE)
                         age_range =fmt_range(AGE)},
           drop = c('SEX', 'AGEGRP', 'AGEGRPN', 'SAFFL',
                   'RACE', 'SEX_FMT', 'RACE_FMT'),
           {
             VAR = 'AGE'
           }) %>
  proc_sort(by = v(VAR),
            keep = v(VAR, age_n, age_mean_sd, age_median,
                     age_quantile, age_range),
            options = 'nodupkey')
```

USUBID	AGE	TRTGRP1	TRTGRP1N	age_range	age_quantile	age_median	age_n	age_mean_sd	VAR
1 00003	21	Placebo	1	11 - 25	16.0 - 20.0	18.0	100	17.8 (3.0)	AGE
2 00004	22	Placebo	1	11 - 25	16.0 - 20.0	18.0	100	17.8 (3.0)	AGE
3 00005	19	Placebo	1	11 - 25	16.0 - 20.0	18.0	100	17.8 (3.0)	AGE
4 00009	21	Placebo	1	11 - 25	16.0 - 20.0	18.0	100	17.8 (3.0)	AGE
5 00010	19	Placebo	1	11 - 25	16.0 - 20.0	18.0	100	17.8 (3.0)	AGE

VAR	age_n	age_mean_sd	age_median	age_quantile	age_range
1 AGE	100	17.8 (3.0)	18.0	16.0 - 20.0	11 - 25

Only keep the unique records according by option

formats()-Example



Deep Dive in reporter

- `add_content`: Adds an object to the report content list. A report will accept multiple pieces of content.
- `add_style`: Adds a style object to a report specification.
- `cell_style`: A class to define the style for a cell in a table.
- `column_defaults`: A function to set default attributes for columns on a table, like column statement in SAS.
- `create_plot`: Create a plot specification that can be added as content to a report. The function supports plot objects returned by ggplot or ggsurvplot. It does not support the Base R plot function.
- `create_report`: Creates a report shell to which you may add titles, footnotes, content, etc
- `create_style`: Creates a style object to control background colors and font settings on your report. The style object can be applied to a report using the `add_style()` function. Currently, styles may only be applied to HTML reports.
- `create_table`: The `create_table` function creates a table object to which further specifications can be added. The object can be added to a report using the `add_content` function.
- `define`: A function to define a table column.
- `footnotes`: Adds one or more footnotes to the report.
- `lowercase_parens`: These functions are used to format the "N=" population label on column headers.
- `page_by`: Adds a page by variable to a report, table, or plot, which will generate a page break for each value of the page by variable.

Deep Dive in reporter

- `page_footer`: Adds a page footer to the report. The page footer will appear on each page of the report, at the bottom of the page.
- `page_header`: Adds a page header to the report. The page header will appear at the top of each page of the report.
- `set_margins`: Sets the page margins for the report. The units for this parameter can be inches or centimeters, depending on the units of measure specified on the `create_report` function.
- `spanning_header`: Creates a header that spans multiple columns. Spanning headers are used to group related columns.
- `stub`: Combines columns into a nested report stub.
- `titles`: Adds one or more titles to an object as a title block.
- `write_report`: This function writes a `report_spec` object to the file system, using the specifications provided in the object.

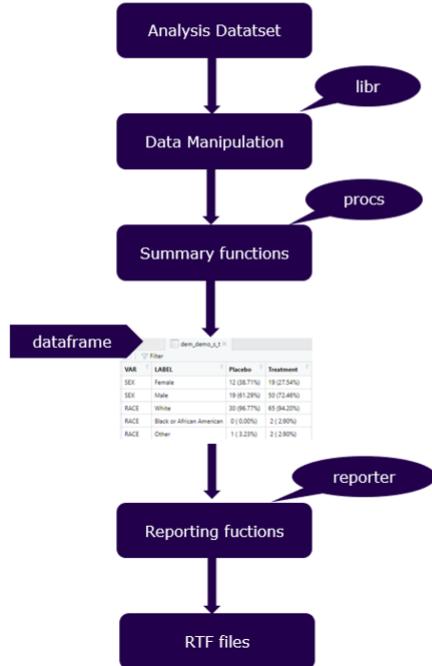
Summary

sanofi

Summary

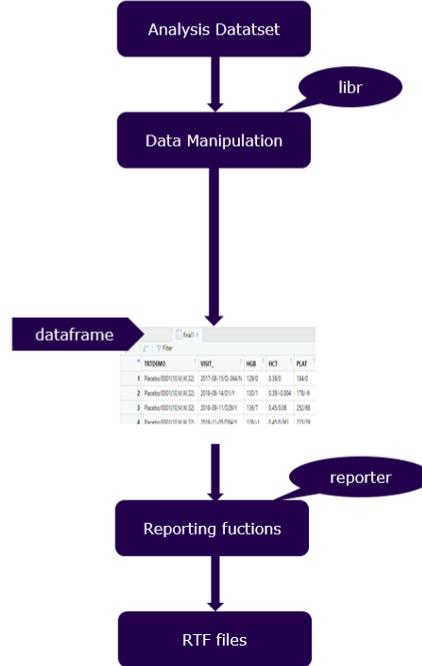
Table:

Summarize the data and store in a dataframe; pass dataframe through the reporting functions.



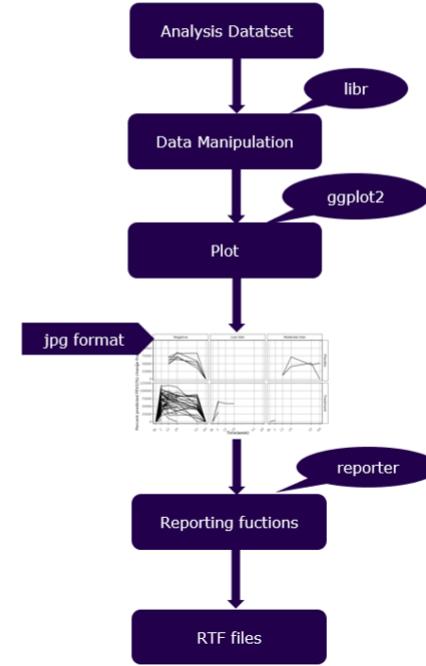
Listing:

Gather the data and store in a dataframe; pass dataframe through the reporting functions.



Graphs:

Summarize the data and plot with ggplot2 or ggsurvfit. Pass the plot through the reporting functions.



References

- [sassy: vignettes/sassy-dm.Rmd \(rdrr.io\)](#)
- [CRAN - Package sassy \(r-project.org\)](#)
- [Application of R language in clinical data \(lexjansen.com\)](#)

Q&A

sanofi

•
Thank you
•

sanofi