PharmaSUG China 2023 - Paper DV-105

Application of R language in clinical data

Perphy ZHAO, Sanofi;
Melody YU, Q2BI;
Michael WANG, Sanofi

ABSTRACT

R is a shared, ideal and popular programming language for statistical analysis and data visualization which provides a flexible and interactive environment and various packages to clean, tidy and analyze data, etc. for everyone related to mathematics and statistics fields. Therefore, no exception for biostatisticians and programmers in the pharmaceutical and biotech industries, it provides a wide and stable user-developed packages which can be efficiently manipulated complex datasets such as SDTM and ADaM datasets and create TLFs. While SAS still remains a critical tool in these industries, the popularity of R in any fields related to data analysis increased exponentially in the past decade due to its many good performances such as open-source, powerful statistical support and advanced visualization. This paper introduces a pragmatic application of R by describing an actual use case of clinical trial data manipulation and creation of tables and figures.

INTRODUCTION

In May of 2015, FDA published a "Statistical Software Clarifying Statement", which contained the following text:

FDA does not require use of any specific software for statistical analyses, and statistical software is not explicitly discussed in Title 21 of the Code of Federal Regulations [e.g., in 21CFR part 11]. However, the software package(s) used for statistical analyses should be fully documented in the submission, including version and build identification.

As noted in the FDA guidance, E9 Statistical Principles for Clinical Trials (available at http://www.fda.gov/Drugs/GuidanceComplianceRegulatoryInformation/Guidances/default.htm), "The computer software used for data management and statistical analysis should be reliable, and documentation of appropriate software testing procedures should be available." Sponsors are encouraged to consult with FDA review teams and especially with FDA statisticians regarding the choice and suitability of statistical software packages at an early stage in the product development process.
With the development of science and technology, R has been a powerful programming language in all walks of life. On one hand, there have been many pieces of paper or presentation by powerful R in supporting innovative statistical methodologies and advanced visualization. On the other hand, as a statistical programmer working in pharma industry, it costs much time to create SDTM, ADaM datasets and TFLs which have been predominantly done using SAS in our day-to-day work. There is little discussion on how to create standard datasets and outputs by R in pharma industry. In this paper, we will share our experience in exploring R in generating the standard datasets and TFLs which usually done by SAS. As the general data analysis, clinical data goes through the same process as the following process in figure 1.

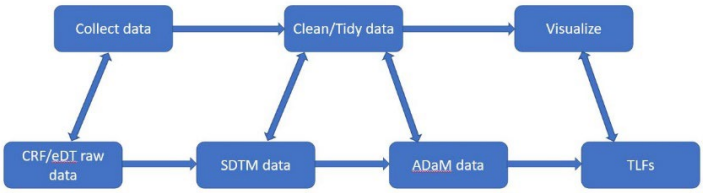


Figure 1. Flow chart

CREATE DATASET

The Most same programming logic exists among the different programming languages, so do R and SAS. Therefore, we can also use R to generate a SDTM dataset following the steps in SAS.

Some explanation needs to be clarified before starting. Hybrid data process is that raw data is mapped into the CDISC standard templates, which only contain the variables from CRF or eDT, after that, we need generate the full SDTM datasets which contain the derived variables needed as per the study. The purpose of doing this is to reduce the difficulty and risk of from raw data to full SDTM directly.

SDTM

1. Initial the program and clean the temporary objects by rm.all() function.

```
rm.all(except = list.GlobalEnv)
```

2. Loading the packages using the library() function which is different from libname in SAS. But if you never installed the packages you need to use, you should install these packages using install.packages() function firstly.

```
# install.packages('diffdf')
library(tidyverse)
library(diffdf)
```

3. Read metadata or specification file using read_sas() function and get the variables list. 'PMETSDTM' is the physical path, which is used to store SAS datasets or other files such as the metadata or specification file.

```
# read the metadata
domain <- 'EX'
md_variable <- read_sas(paste(PMETSDTM, 'variable.sas7bdat', sep = '/'))
md_dataset <- read_sas(file.path(PMETSDTM, 'dataset.sas7bdat'))
md_supp <- read_sas(file.path(PMETSDTM, 'supp.sas7bdat'))
# list of parent domain variables
var_sdtm_parent <- with(md_variable, VARIABLE[DOMAIN == domain])
# list of suppqual variable names
var_suppqual <- md_variable$VARIABLE[md_variable$DOMAIN ==
                                       paste('SUPP', domain, sep = '')]
kp_supp <- md_supp %>%
  filter(DOMAIN == paste('SUPP', domain, sep = ''))
# list of key variables
key_var <- filter(md_dataset, PARENT == domain) %>%
  select(KEYS)
# list of non-hybrid variables
nhybrid_var <- md_variable %>%
  filter(PARENT == domain & SDTM_HYB != 'Y') %>%
  select(VARIABLE)
# keep vars
keep_var <- md_variable %>%
  filter(., PARENT == domain ) %>%
  mutate(ORDNUM1 = as.numeric(ORDNUM)) %>%
  arrange(ORDNUM1) %>%
  select(VARIABLE)
```

4. Pre-process which is to do for deriving non-hybrid variables, such as translate the null value into NA for character variables, import the external files which are needed to derive new variables, etc.

```
# load the external functions used the program
source('function.R')
lead <- read_sas(paste(SDD, 'ex_f.sas7bdat', sep = '/'))
# read hybrid data
ex_h <- read_sas(file.path(RDBS, 'ex_h.sas7bdat')) %>%
  mutate(across(where(is.character),
```

```
                ~ if_else(! .x %in% c('', '.'),
                          .x,
                          NA_character_
                         )
               )
        )
dm <- read_sas(file.path(SDD, 'dm_f.sas7bdat')) %>%
  mutate(across(where(is.character),
                ~ if_else(! .x %in% c('', '.'),
                          .x,
                          NA_character_
                         )
               ),
         RFSTDTC = case_when(
                           ENROLLED == 'Y' & SAFETY == 'N' & is.na(RFSTDTC)
                           ~ ENROLDTC,
                           TRUE ~ RFSTDTC
                          )
        ) %>%
  select(USUBJID, RFSTDTC, RFENDTC, RFPENDTC, TEENDTC, ARMCD, ACTARMCD)
# get actual arm information from ex and rnd_study
rnd <- read_sas(file.path(ADD, 'rnd_study.sas7bdat')) %>%
  mutate(across(where(is.character),
                ~ if_else(! .x %in% c('', '.'),
                          .x,
                          NA_character_
                         )
               ),
         exrefid = as.character(extrtnb)
        )
```

5. Derive new variables which are needed to show for the study and keep all variables in datasets. Figure 2 is shown partial variables of EX domain.

```
ex1 <- ex_h %>%
  left_join(dm, by = c('USUBJID')) %>%
  left_join(rnd, by = c('EXREFID' = 'exrefid')) %>%
  mutate(exivfl = if_else(EXREFID == as.character(extrtnb),1,0),
         dose_ = str_split(trtgrp_lglabel, pattern = ' ', simplify = TRUE),
         EXDOSE = as.numeric(str_extract(dose_[,2], '\\d + ')),
         EXDOSU = str_extract(dose_[,2], '[a-z] + '),
         EXDOSFRM = 'INJECTION',
         EXCAT = 'INVESTIGATIONAL PRODUCT ADMINISTRATION',
         EPOCH = pmap_chr(.l = list(RFSTDTC, TEENDTC, RFPENDTC, RFENDTC,
                                    EXSTDTC),
                          .f = ~epoch(RFSTDTC = ..1, TEENDTC = ..2,
                                      RFPENDTC = ..3, RFENDTC = ..4,
                                      DTVAR = ..5, ds = 'EX'
                                     )
                         ),
         VISIT = if_else(VISITNUM == 9900, 'UNSCHEDULED VISIT', VISIT),
         EXSTDY = as.integer(pmap_chr(.l = list(EXSTDTC, RFSTDTC),
                                      .f = ~day(DTC = ..1, RFSTDTC = ..2)
                                     )
                             ),
         EXENDY = pmap_int(.l = list(EXENDTC, RFSTDTC),
                           .f = ~day(DTC = ..1, RFSTDTC = ..2)
                          )
```

```
      ) %>%
  arrange(key_var[[1]][[1]]) %>%
  group_by(USUBJID) %>%
  mutate(EXSEQ = 1:n()) %>%
  ungroup() %>%
  select(c(keep_var[[1]]))
```

EXSTDTC Start Date/Time of Treatment	EXENDTC End Date/Time of Treatment	EXSTDY Study Day of Start of Treatment	EXENDY Study Day of End of Treatment	EPOCH Epoch
2022-10-03T17:10	2022-10-03T17:10	1	1	TREATMENT
2022-10-17T16:35	2022-10-17T16:35	15	15	TREATMENT
2022-10-31T16:20	2022-10-31T16:20	29	29	TREATMENT
2022-11-14T16:50	2022-11-14T16:50	43	43	TREATMENT
2022-11-29T15:41	2022-11-29T15:41	58	58	TREATMENT
2022-12-15T15:35	2022-12-15T15:35	74	74	TREATMENT
2022-12-26T15:50	2022-12-26T15:50	85	85	TREATMENT
2023-01-10T15:20	2023-01-10T15:20	100	100	TREATMENT
2023-01-26T16:22	2023-01-26T16:22	116	116	TREATMENT
2023-02-10T12:21	2023-02-10T12:21	1	1	TREATMENT

Figure 2. Partial variables in ex domain

6. Add label, length to variables by using xportr_**() functions.

```
md_dataset1 <- md_dataset %>%
  rename_with(~tolower(.x)) %>%
  mutate(dataset = tolower(parent))%>%
  filter(dataset %in% c('ex', 'suppex'))
md_variable1 <- md_variable %>%
  rename_with(~tolower(.x)) %>%
  mutate(dataset = tolower(parent),
         length = as.integer(s_length)
        ) %>%
  filter(dataset == 'ex') %>%
  select(dataset, variable, label, length)
ex <- ex1
ex_f <- xportr_df_label(ex, md_dataset1) %>%
  xportr_length(md_variable1) %>%
  xportr_label(md_variable1, domain = 'ex')
map(ex_f, attr, 'label')
## $STUDYID
## [1] "Study Identifier"
## $DOMAIN
## [1] "Domain Abbreviation"
…
attr(ex_f, 'label')
## [1] "Exposure"
map(ex_f, attr, 'width')
## $STUDYID
## [1] 200
## $DOMAIN
## [1] 200
…
```

Where the … presents that omits some result or variables in this paper.

7. Generate parent and supplemental domain.

```
ex <- ex_f %>%
  select(var_sdtm_parent)
```

```
suppex <- ex_f %>%
  select(c(STUDYID, USUBJID, DOMAIN, EXSEQ),var_suppqual) %>%
  pivot_longer(
               -c(STUDYID, USUBJID, DOMAIN, EXSEQ),
               names_to = 'QNAM',
               values_to = 'QVAL',
               values_drop_na = TRUE
              ) %>%
  mutate(RDOMAIN = DOMAIN,
         IDVAR = 'EXSEQ',
         IDVARVAL = EXSEQ,
         dataset = 'SUPPEX'
        ) %>%
  left_join(md_variable, by = c('dataset' = 'DOMAIN','QNAM' = 'VARIABLE'))
%>%
  mutate(QLABEL = LABEL,
         QORIG = ORIGIN,
         QEVAL = EVALU
        ) %>%
  select(kp_supp[['VARIABLE']])
md_supp1 <- md_supp %>%
  rename_with(~tolower(.x)) %>%
  mutate(
         dataset = tolower(domain),
         length = as.integer(s_length)
        ) %>%
  filter(dataset == 'suppex') %>%
  select(dataset, variable, label, length)
suppex <- suppex %>%
  xportr_df_label(md_dataset1) %>%
  xportr_label(md_supp1) %>%
  xportr_length(md_supp1)
```

8. Validate the developed dataset if you are a QCer.

```
lead1 <- lead %>%
  mutate(across(where(is.character),
                ~ if_else(! .x %in% c('', '.'),
                          .x,
                          NA_character_
                         )
               ),
         across(c(EXSEQ, EXSTDY),
                ~ as.integer(.x)
               )
        )
dif <- diffdf(ex_f, lead1, suppress_warnings = TRUE)
dif$AttribDiffs
## # A tibble: 43 × 4
##    VARIABLE ATTR_NAME VALUES.BASE VALUES.COMP
##  * <chr>    <chr>     <list>      <list>
##  1 DOMAIN   width     <dbl [1]>   <NULL>
##  2 DOMAIN   label     <chr [1]>   <NULL>
## # … with 33 more rows
dif$NumDiff
## NULL
dif$VarDiff_EPOCH
## NULL
```

ADaM

We can develop ADaM datasets as the same steps to develop SDTM, therefore, the steps to ADaM are omit this section.

CREATE TLF

TABLE

1. Create the dataset for reporting as in SAS.

```
library(tidyverse)
library(sassy)
# import SAS data
adsl <- haven::read_sas(file.path(ADDM, 'adsl.sas7bdat'))
# Analysis Data Preparation and Manipulation
trt_vars <- c('TRTGRP1')
trt_varns <- trt_vars %>% paste0('N')
demo_cate_vars <- c('AGEBLGR', 'AGEGR1', ……)
demo_cate_varns <- demo_cate_vars %>% paste0('N')
demo_cont_vars <- c('AGEBL', 'AGE', 'HGTBL', 'WGTBL', 'BMIBL')
# prepare data
anl1_adsl <- adsl %>%
  mutate(SEXN = match(SEX, c('Male', 'Female')),
         ETHNICN = match(ETHNIC,
                         c('Hispanic or Latino', 'Not Hispanic or Latino')
                        )
        ) %>%
  select(STUDYID, USUBJID, SAFFL,
         one_of(c(trt_vars, trt_varns, demo_cont_vars, demo_cate_vars,
                  demo_cate_varns)
               )
        ) %>%
  filter(SAFFL == 'Y')
# get ARM population counts
arm_pop <- count(anl1_adsl, TRTGRP1) %>%
  deframe()
# create summary statistics for baseline character
all_cont_block <- anl1_adsl %>%
  group_by(TRTGRP1) %>%
  summarise(
            # for decimal = 0
            across(c(AGEBL, AGE),
                   list(`Number` = ~fmt_n(.),
                        `Mean (SD)` = ~fmt_mean_sd(., format = '%.1f'),
                        `Median` = ~fmt_median(.),
                        `Q1 ; Q3` = ~fmt_quantile_range(., sep = ';',
                                                        format = '%.1f'),
                        `Min ; Max` = ~fmt_range(.,sep = ';', format = '%.f')
                       )
                  ),
            # for decimal = 1
            across(c(HGTBL, WGTBL, BMIBL),
                   list(`Number` = ~fmt_n(.),
                        `Mean (SD)` = ~fmt_mean_sd(., format = '%.2f'),
                        `Median` = ~fmt_median(.),
                        `Q1 ; Q3` = ~fmt_quantile_range(., sep = ';',
```

```
                                                        format = '%.2f'),
                        `Min ; Max` = ~fmt_range(.,sep = ';', format = '%.1f')
                      )
                  )
           ) %>%
  pivot_longer(-TRTGRP1, names_to = 'label', values_to = 'value') %>%
  pivot_wider(names_from = TRTGRP1, values_from = 'value') %>%
  add_column(var = '', .before = 'label') %>%
  datastep({label2 <- str_split(label, '_')}) %>%
  unnest_wider(label2) %>%
  mutate(var = ...1, label = ...2) %>%
  select(-c(...1, ...2))
# create category group frequency counts
cate_block <- anl1_adsl %>%
  select(USUBJID, AGEBLGR, SEX, REGION1, ETHNIC, WTBLGR1, BMIGR1,
         TRTGRP1) %>%
  datastep(
           {
            if (AGEBLGR != ''){ITEM = 'AGEBLGR'
                               ITEMV = 'Number'
                               output()
                               ITEMV = AGEBLGR
                               output()
                              }
            …
           }
          ) %>%
  group_by(TRTGRP1, ITEM, ITEMV) %>%
  summarize(n = n(), .groups = 'keep')
var_fmt <- c('AGEBL' = 'Baseline age (years)',
             'AGEBLGR' = 'Baseline age group [n (%)]', …)
var_ord <- c('AGEBL', 'AGEBLGR', …)
denom <- cate_block %>%
  datastep(keep = c('TRTGRP1', 'ITEM', 'n'),
           rename = c(n = 'denom'),
           {if (ITEMV == 'Number')
            output()}, drop = c('ITEMV'))
cate_block1 <- cate_block %>%
  datastep(merge = denom,
           merge_by = c('TRTGRP1', 'ITEM'),
           merge_in = c('inA', 'inB'),
           {
            if (n == denom) value = paste(n,' (100%)')
            else if (n == 0) value = paste(n)
            else value = paste(n,'(',round(n/denom*100,0),'%)')
            if (ITEMV == 'Number') value = paste(n)
            ITEM = as.factor(ITEM)
            ITEMV = as.factor(ITEMV)
           }
          ) %>%
  select(TRTGRP1, ITEM, ITEMV, value) %>%
  group_by(TRTGRP1, ITEM, ITEMV) %>%
  pivot_wider(names_from = TRTGRP1,
              values_from = value,
              values_fill = '0'
             ) %>%
  rename(c(var = ITEM, label = ITEMV))
```

```
# combine all blocks into final data frame
final <- bind_rows(all_cont_block, cate_block1) %>%
  datastep(
           {
            ord2 = n.
            if (var %in% demo_cate_vars & label == 'Number') ord2 = -999
            ord1 = match(var, var_ord)
           }
          ) %>%
  arrange(ord1, ord2) %>%
  select(-contains('ord'))
```

Part of the final data used to create TLF is shown as figure 3. Age and age group at baseline are only shown here.

var	label	Placebo	Treatment
AGEBL	Number	80	162
AGEBL	Mean (SD)	9.3 (1.7)	9.1 (1.8)
AGEBL	Median	10.0	10.0
AGEBL	Q1 ; Q3	8.0 ; 11.0	8.0 ; 11.0
AGEBL	Min ; Max	6 ; 11	6 ; 11
AGEBLGR	Number	80	162
AGEBLGR	[6-8]	28 (35 %)	68 (42 %)
AGEBLGR	[9-11]	52 (65 %)	94 (58 %)

Figure 3. Partial data for reporting

2. Create table using create_table() function.

```
# Create Table
tbl <- create_table(final, first_row_blank = TRUE, continuous = FALSE,
                    borders = c('top', 'bottom')
                   ) %>%
  column_defaults(vars = c('var', 'label', `Placebo`, `Treatment`),
                  align = 'center', width = 3) %>%
  stub(vars = c('var', 'label'), '', width = 3) %>%
  define(var, blank_after = TRUE, dedupe = TRUE, label = '',
         format = var_fmt,label_row = TRUE, page_break = FALSE) %>%
  define(label, indent = .25, label = 'Demographic Category') %>%
  define(`Placebo`, label = 'Placebo', n = arm_pop['Placebo']) %>%
  define(`Treatment`, label = 'Treatment', n = arm_pop['Treatment'])
```

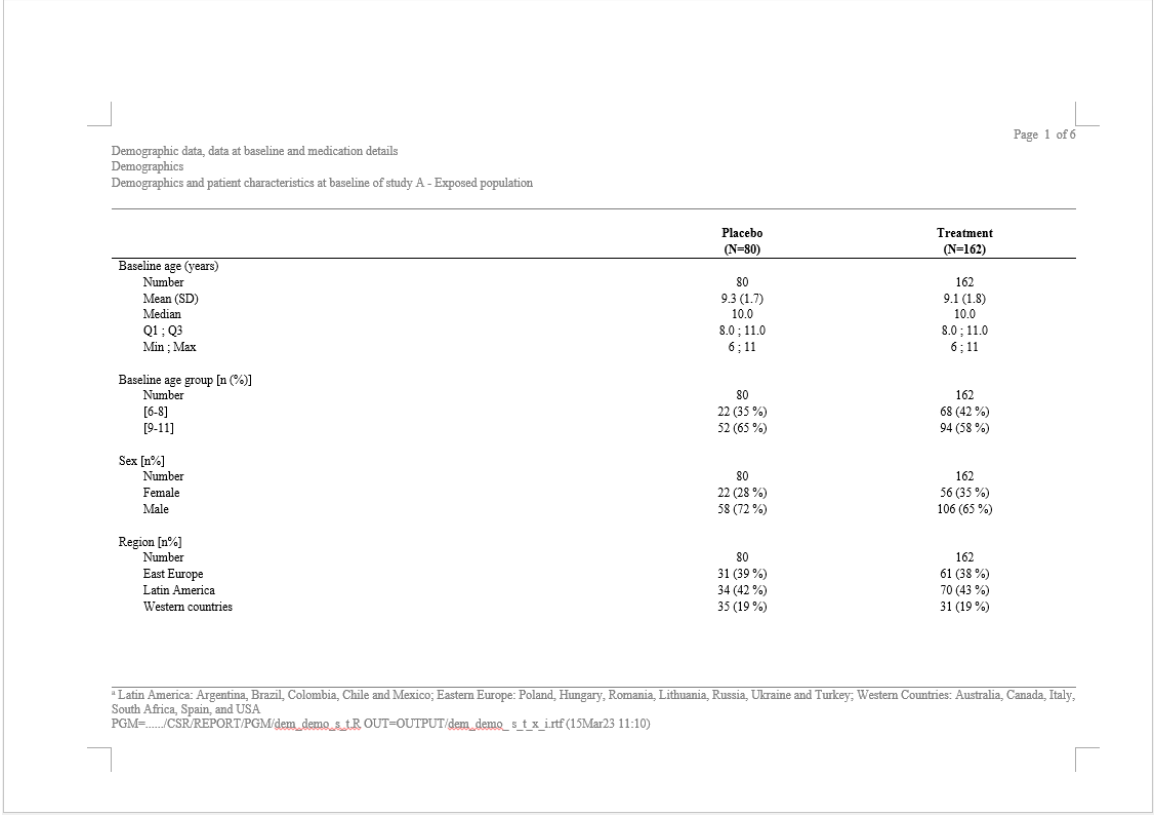
3. Create RTF using create_report() function.

```
# Create RTF
pth <- file.path(MISC, 'R/Lead/dem_demo_s_t.rtf')

rpt <- create_report(pth, output_type = 'RTF', font = 'Times') %>%
  set_margins(top = 1, bottom = 1, left = 0.5, right = 0.5) %>%
  page_header(c('', 'Demographic data, data at baseline and medication
                details', 'Demographics', 'Demographics and patient
                characteristics at baseline of study A - Exposed
                population'),
              'Page [pg] of [tpg]'
```

```
            ) %>%
  add_content(tbl) %>%
  footnotes(c('{\\super a}Latin America: Argentina, Brazil, Colombia, Chile
              and Mexico; Eastern Europe: Poland, Hungary, Romania,
              Lithuania, Russia, Ukraine and Turkey; Western Countries:
              Australia, Canada, Italy, South Africa, Spain, and USA',
              paste0('PGM = …/CSR/REPORT/PGM/dem_demo_s_t.R
                     OUT = OUTPUT/dem_demo_s_t_x.rtf (',
                     fapply(Sys.time(), '%d%b%y %H:%M'), ')'
                    )
             ),
            footer = TRUE, blank_row = 'none'
           )
write_report(rpt)
```

Partial contents of the final TLF report is shown as figure 4. For TLF report, we can also use the r2rtf package to generate the output. For the titles and footnotes, we can read them from the excel file into vectors separately.



Page 1 of 6

Demographic data, data at baseline and medication details
Demographics
Demographics and patient characteristics at baseline of study A - Exposed population

	Placebo (N=80)	Treatment (N=162)
Baseline age (years)		
Number	80	162
Mean (SD)	9.3 (1.7)	9.1 (1.8)
Median	10.0	10.0
Q1 ; Q3	8.0 ; 11.0	8.0 ; 11.0
Min ; Max	6 ; 11	6 ; 11
Baseline age group [n (%)]		
Number	80	162
[6-8]	22 (35 %)	68 (42 %)
[9-11]	52 (65 %)	94 (58 %)
Sex [n%]		
Number	80	162
Female	22 (28 %)	56 (35 %)
Male	58 (72 %)	106 (65 %)
Region [n%]		
Number	80	162
East Europe	31 (39 %)	61 (38 %)
Latin America	34 (42 %)	70 (43 %)
Western countries	35 (19 %)	31 (19 %)

[a] Latin America: Argentina, Brazil, Colombia, Chile and Mexico; Eastern Europe: Poland, Hungary, Romania, Lithuania, Russia, Ukraine and Turkey; Western Countries: Australia, Canada, Italy, South Africa, Spain, and USA
PGM=...../CSR/REPORT/PGM/dem_demo_s_t.R OUT=OUTPUT/dem_demo_ s_t_x_i.rtf (15Mar23 11:10)

Figure 4. Partial demographic output

FIGURE

There are many geom_**() functions in ggplot2 package that we can use them to plot most figures such bar, histogram, box plots, etc. This section will show how to create the error bar plot.

Figure 5 is the data for change from baseline plot. We create this plot by geom_line() and geom_errorbar() functions here and we only show the placebo group.

TRTGRP1	lbvisitn	lbvisit	mean	up	low
Placebo	-6	BL	0.0000000	0.000000	0.000000
Placebo	0	Week 0	0.3003109	10.172780	-7.199700
Placebo	4	Week 4	-2.1935073	8.472637	-10.487193
Placebo	12	Week 12	-1.7590576	8.762422	-9.908079
Placebo	24	Week 24	-0.7143440	9.493546	-8.549776
Placebo	36	Week 36	-0.9017792	9.198343	-8.629444
Placebo	52	Week 52	-0.2267835	9.964636	-8.045744
Placebo	64	Week 64	1.4094647	11.235000	-7.118939

Figure 5. data for change from baseline plot

1. Create the plot.

```
library(ggplot2)
plot1 <- ggplot(configs$data, aes(x = lbvisitn, y = mean)) +
  geom_line(aes(group = TRTGRP, color = TRTGRP, linetype = TRTGRP),
            size = 1,
            position = position_dodge(1.5)
           ) +
  scale_color_manual(values = values = c('red', 'blue')) +
  geom_errorbar(aes(ymin = low, ymax = up, color = TRTGRP),
                position = position_dodge(1.5),
                width = 0.5,
                size = 1
               ) +
  geom_point(aes(color = TRTGRP, shape = TRTGRP),
             size = 2,
             position = position_dodge(1.5)
            ) +
  scale_shape_manual(values = c(8, 2)) +
  labs(title = '', y = 'Mean Change + /- SE', x = '', fill = '') +
  scale_y_continuous(limits = c(-12, 12),
                     breaks = seq(-12, 12, by = 2),
                     expand = expansion(mult = 0, add = 2)
                    ) +
  theme_classic() +
  theme(plot.title = element_text(color = 'black', size = 15, hjust = 0.5,
                                  vjust = 2),
        legend.position = 'top',
        axis.title.y = element_text(color = 'black', size = 13,
                                    margin = margin(r = 5)),
        axis.title.x = element_text(color = 'black', size = 13,
                                    margin = margin(t = 5)),
        axis.text = element_text(color = 'black', size = 10),
        axis.text.x = element_text(vjust = -1),
        axis.title.x.bottom = element_text(vjust = -2),
        plot.margin = margin(t = 20, r = 10, b = 10, l = 5, unit = 'pt'),
        panel.spacing = unit(0, 'lines'),
```

```
        panel.border = element_blank(),
        strip.background = element_rect(color = 'white'),
        strip.text = element_text(color = 'black', size = 11, vjust = 1,
                                  hjust = 0.5, family = 'sans'),
        strip.placement = 'outside'
       )
```

2. Create the RTF as figure 6.

```
page1 <- create_plot(plot1, 4.5, 7)
rpt <- create_report(file.path(REPO, 'lab_meanchg_hgb_s_g.rtf'),
                     output_type = 'RTF', font = 'Times') %>%
  set_margins(top = 1, bottom = 1) %>%
  page_header(c('', 'Clinical laboratory data', 'Descriptive statistics
                across visits', 'Hemoglobin (g/L): Mean change from baseline
                - Exposed population'),
              'Page [pg] of [tpg]') %>%
  add_content(page1) %>%
  page_footer(paste0('PGM = …/CSR/REPORT/PGM/lab_meanchg_s_g.R
                      OUT = OUTPUT/lab_meanchg_hgb_s_g_x.rtf (',
                      fapply(Sys.time(), '%d%b%y %H:%M'), ')'),
              footer = TRUE,
              blank_row = 'none'
             )
write_report(rpt)
```

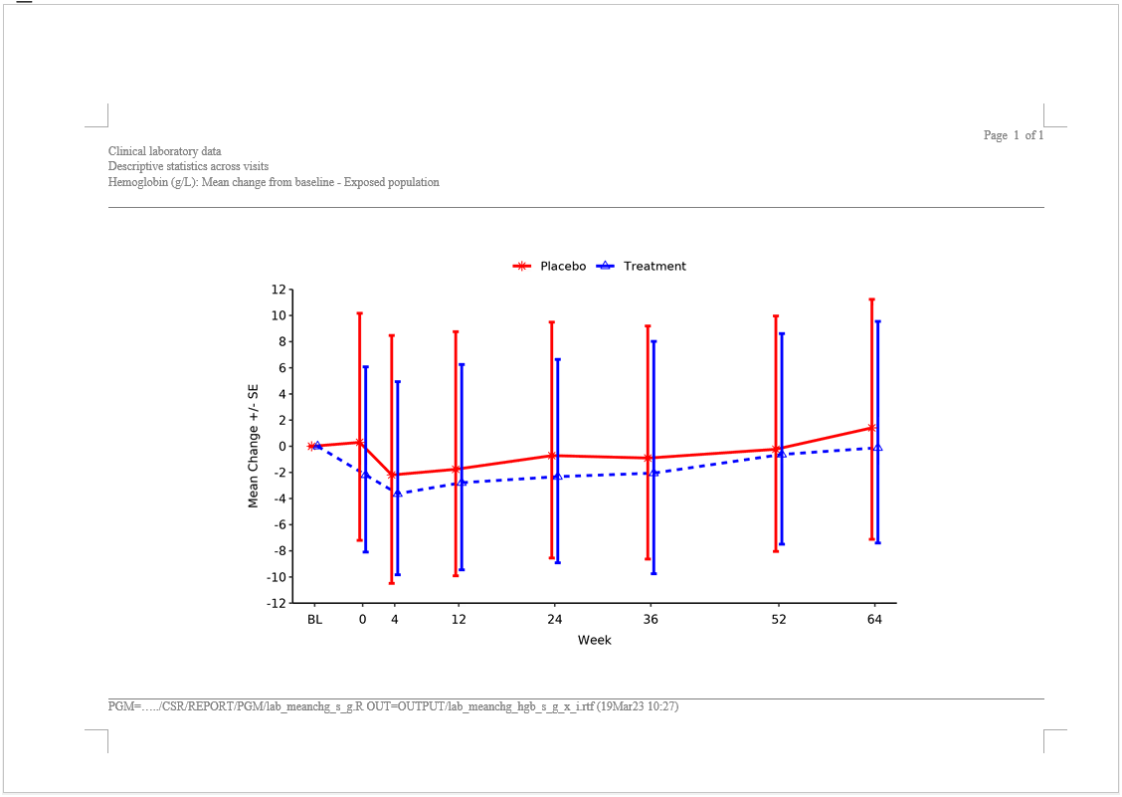


Figure 6. Figure in RTF

RECOMMENDATION

If you are a pure SAS programmer, it's recommended that you can begin to learn R by using libr package whose grammar is more similar with SAS.

```
library(tidyverse)
library(sassy)
# Example: Simple Data Step
df <- datastep(class,
               where = expression(Name != 'Alfred'),
               keep = c('Name', 'Sex', 'Weight', 'wgtgrp', 'wgtgrpn',
                        'class', 'grp', 'num', 'mwgt'),
               retain = list(num = 1),
               calculate = {mwgt = mean(Weight)},
               steps = {if (Weight >= 100) {wgtgrp <- 'High'
                                            wgtgrpn <- 1
                                           }
                        else {wgtgrp <- 'Low'
                              wgtgrpn <- 2
                             }
                        class <- 'class 1'
                        if (n. < 10) grp <- 'group 1'
                        else grp <- 'group 2'
                        if (first.) num <- 1
                        else num + 1
                       }
              ) %>%
  arrange(Sex) %>%
  datastep(by = 'Sex',
           Retain = list(num1 = 1),
           steps = {if (first.) num1 = 1
                    else num1 = num1 + 1
                    if (n. == 1) delete()
                    output()
                    grp = 'TOT'
                    output()
                   }
          )
```

APPENDIX

Function.R

```
# Function for populating the EPOCH
epoch <- function(RFSTDTC, TEENDTC, RFPENDTC, RFENDTC, DTVAR, ds){
  if (ds %in% c('AE', 'CE', 'CM', 'EX', 'EC', 'FAAE')){
    if (! is.na(DTVAR)) {
      if (DTVAR < RFSTDTC | is.na(RFSTDTC)){'SCREENING'}
      else if (RFSTDTC <= DTVAR & (DTVAR <= RFENDTC | is.na(RFENDTC)))
      {'TREATMENT'
      else if (RFENDTC < DTVAR & (DTVAR <= TEENDTC | is.na(TEENDTC)))
      {'RESIDUAL-TREATMENT'}
      else if (RFENDTC < DTVAR & (DTVAR <= RFPENDTC | is.na(RFPENDTC)))
      {'POST-TREATMENT'}
      else if (RFENDTC < DTVAR & ! is.na(RFPENDTC)) {'POST-TREATMENT'}}
    else{NA_character_}}
  else {
    if (! is.na(DTVAR)){
      if (DTVAR <= RFSTDTC | is.na(RFSTDTC)){'SCREENING'}
      else if (RFSTDTC < DTVAR & (DTVAR <= RFENDTC | is.na(RFENDTC)))
      {'TREATMENT'}
      else if (RFENDTC < DTVAR & (DTVAR <= TEENDTC | is.na(TEENDTC)))
```

```
      {'RESIDUAL-TREATMENT'}
      else if (RFENDTC < DTVAR & (DTVAR <= RFPENDTC | is.na(RFPENDTC)))
      {'POST-TREATMENT'}
      else if (RFENDTC < DTVAR & ! is.na(RFPENDTC)) {'POST-TREATMENT'}}
    else{NA_character_}}}
# Function for populating the analysis day
day <- function(DTC, RFSTDTC = RFSTDTC){
  if (DTC >= RFSTDTC) as.Date(DTC, tryFormats = c('%Y-%m-%d', '%Y/%m/%d'))-
                      as.Date(RFSTDTC, tryFormats = c('%Y-%m-%d',
                                                      '%Y/%m/%d')) + 1
  else -as.Date(DTC, tryFormats = c('%Y-%m-%d', '%Y/%m/%d')) +
       as.Date(RFSTDTC, tryFormats = c('%Y-%m-%d', '%Y/%m/%d'))}
```

REFERENCE

[1] Hao Meng. 2020. “R for Clinical Reporting, Yes - Let's Explore It!”. *PharmaSUG 2020 - Paper DV-057*.

[2] Statistical Software Clarifying Statement (fda.gov)

[3] The Comprehensive R Archive Network (r-project.org)

ACKNOWLEGEMENTS

The authors would like to take this opportunity to thank Sanofi and my team for facilitating and supporting this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the authors at:

Perphy ZHAO
perphyzhao@163.com

Melody YU
yjlcq2009@163.com

Michael WANG
michael3.wang@sanofi.com