PharmaSUG 2023 - Paper QT-280

Are you planning to create/validate CDISC data set in R? Here is a step-by-step guide!

Ganeshchandra Gupta, Ephicacy Consulting Group, Inc

ABSTRACT

In recent years, many pharmaceutical companies have adopted R as a data analysis tool. The main reasons for the increasing importance of R are the availability and ever-growing number of high-quality statistical methods, very good graphics capabilities and a wide range of useful programming extensions. However, no single programming language can solve every single problem you will encounter in your programming career. Though SAS® is easy to learn and provides simpler coding options, R on the other hand has a stepwise learning, depending upon programming language. To work with R, by rule of thumb, you will have to know the basics of the R language which is quite easy. And still, no one ever talks about how simple it is to do clinical trial data manipulation and creation of CDISC: SDTM/ADaM data sets. Majority of Pharma/Biotech companies and CROs use double programming technique to validate SAS data sets. And so, we can use R programming to validate data sets created in SAS environment which could potentially reduce the license cost involved. This paper will provide a step-by-step guide on creating and validating SDTM Demographics (DM) domain along with SAS code comparison. Begin from the Beginning!

INTRODUCTION

R is a programming language and an analytics tool that was developed in 1993 by Robert Gentleman and Ross Ihaka at the University of Auckland, Auckland, New Zealand. It is extensively used by Software Programmers, Statisticians, Data Scientists, and Data Miners. It is one of the most popular Data analytics tools used in Data Analytics and Business Analytics. It has numerous applications in domains like healthcare, academics, consulting, finance, media, and many more. Its vast applicability in Statistics, Data Visualization, and Machine Learning have given rise to the demand for certified trained professionals in R. The purpose of this paper is to understand the difference between R and SAS codes at each step. And then create a SDTM Demographics (DM) domain program in R.

CODE COMPARISON OF R VS SAS

#	Procedure/ Functions	SAS Code	R Code
1.	Importing and Reading Dataset	libname rawdata "c:\rawdata"; data dm1; set rawdata.dm; run; ➔ similarly read drug, died, term	**library(haven)** rawdata <- "C:/Users/Ganesh Gupta/Documents/R/R practice/rawdata" dm <- read_sas("rawdata/dm.sas7bdat", NULL) ➔ similarly read drug, died, term
2.	Checking contents of the dataset	proc contents data= dm; run;	**library(Hmisc)** contents(dm)
3.	Assigning old variable values to new variable and hardcoding	studyid= strip(proto); domain= “DM”;	mutate(STUDYID = PROTO, DOMAIN = “DM”)

4.	Creating character variable and using SUBSTR	length subjid $10; subjid= strip(substr(patient,4,4));	SUBJID = as.character(substr(PATIENT,4,7)) Here third attribute “7” is the position of the string. This will extract from 4th position to 7th position.
5.	Concatenating and deriving a new variable	usubjid= catx(“-”, studyid, siteid, subjid);	USUBJID= paste(STUDYID, SITEID, SUBJID, sep="-")
6.	Creating ISO date variable	brthdtc= strip(put(birthdt, yymmdd10.));	**library(parsedate)** BRTHDTC= format_iso_8601(parse_iso_8601(BIRTHDT))
7.	Finding position of the first occurrence of that string’s first character	index_result= index(BRTHDTC, "\\T",);	BRTHDTC= substr(BRTHDTC, 1, regexpr("\\T", BRTHDTC)-1) The regexpr() function gives you the (a) index into each string where the match begins and the (b) length of the match for that string. regexpr() only gives you the *first* match of the string (reading left to right).
8.	Derive age from birth date	age=int((today()-dob)/365.25);	**library(ggplot2)** **library(eeptools)** AGE = floor(age_calc(as.Date(BIRTHDT), as.Date(RANDDT), units="years", precise = TRUE))
9.	If-else statement	if gender= ”FEMALE” then sex= ”F”; else if gender= ”MALE” then sex= ”M”; else sex=””;	SEX1 = ifelse(SEX==”FEMALE”,”F”,ifelse(SEX==”MALE”,”M”,”"))
10.	Sorting a dataset	proc sort data= dm out= sdtm.dm; by studyid usubjid; run;	dm <- arrange(STUDYID, USUBJID)
11.	Remove missing values	data f_dosedtc; set drug1; where not missing (dosedtc); run;	f_dosedtc <- drug1 %>% filter(!is.na(DOSEDTC))
12.	First and last observation by grouping variables	proc sort data=drug1; by subjid dosedt; run; data f_dosedtc l_dosedtc; set drug1;	f_dosedtc <- drug1 %>% group_by(SUBJID) %>% slice_min(order_by = DOSEDT) slice_max(order_by = DOSEDT)

		by subjid; if first.subjid then output f_dosedtc; if last.subjid then output l_dosedtc; run;	
13.	Remove duplicated rows	proc sort data=f_dosedtc out=f_dosedtc1 noduprecs; by subjid; run;	f_dosedtc1 <- unique(f_dosedtc, incomparables=FALSE) #Unique returns a data table with duplicated rows removed
14.	Merge datasets while keeping rows from the left dataset	data dmall; merge dm1(in=indm) f_dosedtc1; by subjid; if indm; run;	dmall <- left_join(dm1, f_dosedtc1, by="SUBJID")
15.	Order and keep only required variables	data dm; retain studyid domain usubjid subjid; set dmall; keep studyid domain usubjid subjid; run;	dm <- dmall %>% select(STUDYID, DOMAIN, USUBJID, SUBJID)

16.	Reset all the labels	proc datasets lib=work memtype=data;
		modify dm;
		attrib _all_ label=' ';
		contents data= work.dm;
		run;
			dm <- remove_all_labels(dm)

17.	Dropping a variable	data dmall; set dmall; drop subjid; run;	dmall <- within(dmall, rm(SUBJID))
18.	Renaming a variable	data dm1; set dm; rename gender=sex; run;	dm <- rename(SEX=SEX1, RACE=RACE1) #new_name = old_name
19.	Output dm.sas7bdat	data sdtm.dm; set work.dm; run;	write.table(dm, "C:/Users/Ganesh Gupta/Documents/R/R practice/sdtm/dm.txt", sep=",", row.names=F, col.names=T)

			In R, you can create either CSV or TXT as output.
20.	Output dm.xpt	libname xportout xport c:\xpt; data xportout.dm; set work.dm; run;	Library(“sasxport”) write.xport(dm, file = "xpt/dm.xpt")
21.	Comparing the datasets	proc compare base= prod compare= qc listall; run;	summary(comparedf(prod, qc)) Usage of summary() provides more detailed summary, similar to listall in SAS

THE INITIAL SETUP

In this section we will go through the preliminary setup that needs to be done when starting to program SDTM DM domain:

```
# Install and Load Packages
install.packages("pacman")
library(pacman)

# By using "pacman::p_load" you can use the p_load function from pacman without actually loading pacman.
pacman::p_load(pacman, dplyr, haven, Hmisc, parsedate, ggplot2, lubridate, eeptools, sjlabelled,
               tidyr, foreign, SASxport)

# Library setup
rawdata <- "C:/Users/Ganesh Gupta/Documents/R/R practice/rawdata"
sdtm    <- "C:/Users/Ganesh Gupta/Documents/R/R practice/sdtm"
xpt     <- "C:/Users/Ganesh Gupta/Documents/R/R practice/xpt"

# Set the current working directory
setwd ('C:/Users/Ganesh Gupta/Documents/R/R practice')

# Read the SAS dataset
dm <- read_sas("rawdata/dm.sas7bdat", NULL)
random <- read_sas("rawdata/random.sas7bdat", NULL)
drug <- read_sas("rawdata/drug.sas7bdat", NULL)
died <- read_sas("rawdata/died.sas7bdat", NULL)
term <- read_sas("rawdata/term.sas7bdat", NULL)
vitals <- read_sas("rawdata/vitals.sas7bdat", NULL)
```

```
# Check contents of the dataset
contents(dm)
```

MAIN PROGRAM

Now that we have identified the initial steps it's time to do data manipulation:

```
# Create variables as per the spec from rawdata DM
dm1 <- dm %>% # read %>% as "and then"
mutate(STUDYID = PROTO,
        DOMAIN="DM",
        SITEID = as.character(substr(SUBJID,1,3)),
        SUBJID1 = as.character(SUBJID),
        USUBJID = paste(STUDYID, SITEID, SUBJID, sep="-"),
        RFICDTC = "",
        BRTHDTC = format_iso_8601(parse_iso_8601(BIRTHDT)),
        BRTHDTC = substr(BRTHDTC, 1, regexpr("\\T", BRTHDTC)-1),
        AGE = floor(age_calc(as.Date(BIRTHDT), as.Date(RANDDT), units="years", precise = TRUE)),
        AGEU = "YEARS",
        SEX1 = ifelse(SEX=="FEMALE","F",ifelse(SEX=="MALE","M","")),
        RACE1 = ifelse(RACE=="BLACK","BLACK OR AFRICAN AMERICAN",
ifelse(RACE=="CAUCASIAN","CAUCASIAN",ifelse(RACE=="ASIAN","ASIAN",""))),
        ETHNIC = ifelse(RACE=="HISPANIC","HISPANIC OR LATINO",""),
        ARMCD = ifelse(TRTGROUP=="Placebo","PBO",ifelse(TRTGROUP=="Active","ACT","")),
        ARM = TRTGROUP,
        ACTARMCD = ARMCD,
        ACTARM = TRTGROUP,
        ARMNRS = "",
        ACTARMUD = "",
        COUNTRY = "",
        DMDTC = "",
        DMDY = "") %>%
select(STUDYID, DOMAIN, USUBJID, SUBJID, SUBJID1, SITEID, RFICDTC,
       BRTHDTC, AGE, AGEU, SEX1, RACE1, ETHNIC, TRTGROUP,
       ARMCD, ARM, ACTARMCD, ACTARM, ARMNRS, ACTARMUD,
       COUNTRY, DMDTC, DMDY) %>%
arrange(SUBJID)

# Create variables as per the spec from rawdata DRUG
drug1 <- drug %>%
```

```
mutate(DOSEDTC = format_iso_8601(parse_iso_8601(DOSEDT)),
          DOSEDTC = substr(DOSEDTC, 1, regexpr("\\+", DOSEDTC)-1)) %>%
select(SUBJID, DOSEDT, DOSEDTC) %>%
arrange(SUBJID, DOSEDTC)

# Get first observation from data DOSEDTC
f_dosedtc <- drug1 %>%
  filter(!is.na(DOSEDTC)) %>% #For non-missing values
  group_by(SUBJID) %>%
  slice_min(order_by = DOSEDT) %>% #For 1st observation
  mutate(RFSTDTC = DOSEDTC, RFXSTDTC = DOSEDTC)%>%
  select(SUBJID, RFSTDTC, RFXSTDTC)

# Unique returns a data table with duplicated rows removed
f_dosedtc1 <- unique(f_dosedtc, incomparables=FALSE)

# Get last observation from data DOSEDTC
l_dosedtc <- drug1 %>%
  filter(!is.na(DOSEDTC)) %>%
  group_by(SUBJID) %>%
  slice_max(order_by = DOSEDT) %>%
  mutate(RFENDTC = DOSEDTC, RFXENDTC = DOSEDTC) %>%
  select(SUBJID, RFENDTC, RFXENDTC)

l_dosedtc1 <- unique(l_dosedtc, incomparables=FALSE)
# Create variables as per the spec from rawdata DRUG
term1 <- term %>%
  mutate(RFPENDTC = format_iso_8601(parse_iso_8601(TERMDT)),
            RFPENDTC = substr(RFPENDTC, 1, regexpr("\\T", RFPENDTC)-1)) %>%
  select(SUBJID, RFPENDTC) %>%
  arrange(SUBJID)

# Create variables as per the spec from rawdata DIED
died1 <- died %>%
  mutate(DTHDTC = format_iso_8601(parse_iso_8601(DEATHDT)),
            DTHDTC = substr(DTHDTC, 1, regexpr("\\T", DTHDTC)-1),
            DTHFL="Y") %>%
```

```
  select(SUBJID, DTHDTC, DTHFL) %>%
  arrange(SUBJID)

# Merge all the datasets to create DM
dmall <- left_join(left_join(left_join(left_join(dm1, f_dosedtc1, by="SUBJID"),
                 l_dosedtc1, by="SUBJID"),
                       term1, by="SUBJID"),
                             died1, by="SUBJID")

# Dropping a variable SUBJID
dmall <- within(dmall, rm(SUBJID))

# Rename variables
dmall <- dmall %>%
  rename(SUBJID=SUBJID1, SEX=SEX1, RACE=RACE1) #new_name = old_name

# Order and keep only required variables in final DM
dm <- dmall %>%
select(STUDYID, DOMAIN, USUBJID, SUBJID, RFSTDTC, RFENDTC, RFXSTDTC, RFXENDTC,
RFICDTC, RFPENDTC, DTHDTC, DTHFL, SITEID, BRTHDTC, AGE, AGEU, SEX, RACE, ETHNIC,
ARMCD, ARM, ACTARMCD, ACTARM, ARMNRS, ACTARMUD, COUNTRY, DMDTC, DMDY)
# Reset all the labels
dm <- remove_all_labels(dm)

# Assign label to dataset
label(dm) <- "Demographics"

# To check if unique subjects are present
check <- dm %>% distinct(SUBJID)

# Below steps done to access SAS dataset from SAS applications
# Function defined for converting factors and blanks
convert_format_r2sas <- function(dm){
  dm <- dm %>%
    dplyr::mutate_if(is.factor, as.character) %>%
    dplyr::mutate_if(is.character, tidyr::replace_na, replace = "")
  return(dm)
}
```

```
# Convert some formatting
dm <- convert_format_r2sas(dm)

# Ensure the data and code files are saved in an easily accessible location
# (ideally in or downstream of your R project directory)
write.table(dm, "C:/Users/Ganesh Gupta/Documents/R/R practice/sdtm/dm.txt",
            sep=",",
            row.names=F,
            col.names=T)

# Output DM.xpt
write.xport(dm, file = "xpt/dm.xpt")
```

CONCLUSION

Focus was on the SAS and R language to create the SDTM DM domain

- SAS comes along with a higher proof of quality.
- R packages implement the state-of-the-art algorithms
- There is no general better or worse.

ACKNOWLEDGMENTS

The author would like to extend their sincere thanks to Ephicacy Consulting Group, Inc for giving them an opportunity to write this paper. Any brand and product names are trademarks of their respective companies.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Ganeshchandra Gupta, (Senior Manager - Biostatistics & Programming), Ephicacy Consulting Group, Inc
Phone: +1-848-219-8131, E-mail: ganeshchandra.gupta@ephicacy.com