PharmaSUG China 2022 - Paper CC-110

ADaM creation: SAS or R?

Yan Qiao, BeiGene

INTRODUCTION

Both SAS and R have great ability in data manipulation, and both can be used for creating ADaM datasets. SAS has been used in industry for years and provided various functions in handling data. In recent years, R has also developed several popular and strong packages and submitting in R has become a future trend of the industry. This paper summarizes publicly available procedures/functions provided by SAS and R for data manipulating in section 1; introduces some self-defined SAS utility macros and R utility functions for ADaM creation in section 2; compares the differences between SAS and R regarding data manipulation in section 3.

SECTION 1 AVAILABLE SAS & R FUNCTIONS

SAS has provided us with a great number of procedures and functions in handling data; even though base R has provided us with limited functions, users have developed other packages with very useful functions which can do the same job as SAS does, such as dplyr, Hmisc, haven, readxl and SASxport. Below is a summary of SAS procedures/functions and R functions:

Functionality	**SAS procedure/function**	**R function(package)**
Read in SAS data	set	read_sas(haven)
Read in excel file	proc import	read_excel(readxl)
Create XPT file	libname xpt and proc copy	write.xport (SASxport)
Convert between character and numeric variables	put, input	as.numeric (base), as.character (base), format(base), formatC(base), as.Date (base), as.POSIXct (base), as.POSIXlt (base)
Extract a substring	substr	substr (base)
Search for matches of a string	find/index/prxmatch	grepl (base)
Replace a string with another string	translate/tranwrd/prxchange	gsub (base)
Remove specified characters from a string	compress/prxchange	gsub(base)
Capture text string between two delimiters	scan/prxchange	gsub (base)
Split a string column into multiple columns	scan	separate (tidyr)

Concatenate multiple columns into one column	cats	paste (base), unite (dplyr)
Sort dataset	proc sort	arrange(dplyr)
Transpose dataset	proc transpose	spread (tidyr) [1]
Merge datasets	merge	left_join (dplyr), right_join (dplyr), inner_join (dplyr), full_join (dplyr)
Combine datasets	set	bind_rows (dplyr)
Remove duplicate records	proc sort with nodup	distinct (dplyr)
Group	by	group_by (dplyr)
Keep/drop some variables	keep/drop	select (dplyr)
To pick some records	if/where	filter (dplyr)
Add new variables		mutate (dplyr), summarise (dplyr)
Shift back by a given number of observations	lag*n*	lag (dplyr)
Shift forward by a given number of observations		lead (dplyr)
Rename variable	rename	rename(dplyr)
Attach variable label	label	label (Hmisc)
Check dataset	proc contents	contents (Hmisc)

[1] In the latest version of tidyr, spread and gather would be superseded by pivot_wider and pivot_longer to transpose data.

As shown above, both SAS and R and do the similar job. SAS is more robust and widely used for regulatory submission; better acceptable by users since most users are more familiar with SAS. However sometimes SAS code is not concise enough (date set must be sorted before merging, procedures must be writing one by one). On the other hand, R is more flexible due to the availability of various data packages and functions; simpler and more intuitive since pipes allows the user to express a sequence of multiple operations. However various data packages and frequent unpensioning introduces the concern of reliability and robustness of R; some r functions may require additional knowledge such as “gsub” require some knowledge of regular expressing; various data types require special attention while manipulating data with r, such as distinguishing between NULL and NA, distinguish between character and logical data.

SECTION 2 SELF-DEFINED SAS MACROS & R FUNCTIONS

Besides the publicly available functions, user can also develop their own SAS macros and R functions for ADaM creation since there are several repeated steps while building ADaM datasets. Below is the list of SAS utility macros and R utility functions that we have developed:

Functionality	**SAS Macro**	**R function**	**Usage Scope**
Read in ADaM specification to gather the information on dataset & variable attributes and apply to the final output dataset.	%m_adamds_read; %m_adamds_write	r_applyspec	All ADaM datasets.
Merge SDTM main domain with supplementary domain;	%m_adamds_sdtmmerge	r_sdtmmerge	All ADaM datasets.
Add core variables from ADSL.	%m_adamds_addcore	r_addcore	All ADaM datasets, except for ADSL.
Derive BASE, BASEC, CHG, PCHG.	%m_adamds_base	r_base	BDS datasets, such as ADLB, ADEG, ADVS, ect.
Derive ASEQ.	%m_adamds_aseq	r_aseq	BDS datasets, such as ADLB, ADEG, ADVS, ect.
Convert character date in ISO8601 format to numeric date and implement imputation.	%m_adamds_dtc2dt	r_dtc2dt	Occurrence datasets where date imputation is needed, such as ADAE, ADCM, ADPR, ect.
Search variables ending with DTC to find the max date for each subject.	%m_adamds_maxdt	r_maxdt	To derive last known alive date for ADSL.
Read in excel codelists file to generate formats and informats.	%m_adamds_formats	Not applicable in R.	1. Generate numeric variables based on character version: agegr1n=input(agegr1, agegr1n.); paramn=input(paramcd, paramn.). 2. Generate character variables based on another character version: cntrytxt=put(country, $country.) param=put(paramcd, $param.)
Derive BOR for RECIST, iRECIST and other criteria.	%m_adamds_bor	r_bor	To derive params in ADEF.

Derive missing 2 consecutive TA flag and other efficacy date variables.	%m_adamds_m2ta	r_m2ta	To derive params in ADEF which will be further used in ADTTE.
Generate stand-alone SAS programs without macros.	%m_adamds_demacro_pre; %m_adamds_demacro_post;	Not applicable in R.	All ADaM datasets.
Resize length of all character variables to maximum length required	%m_adamds_resize	Not applicable in R.	All ADaM datasets.
Convert SAS datasets to XPT file under a directory	%m_xpt	Not applicable in R. XPT file is the direct output of each R program.	All ADaM datasets.

As shown above, users can develop tools to support ADaM creation with both SAS and R. They are developed to cover the frequently repeated steps while developing ADaM datasets. A robust SAS macro usually contains the following parts: parameter validation, check and return warning/error message in the log, restore environment. In R function, we can also validate input argument and print warning message. However, developers need to keep in mind that R is not a macro language and is vastly different from SAS. There is no equivalent of SAS macro variables in R. Developers need to take the advantage of R and find alternative solutions.

SECTION 3 COMPARISON BETWEEN SAS & R

By using the available SAS procedures/functions & R functions as well as self-define SAS utility macros and R utility functions, we have successfully created ADaM datasets with both SAS and R and have compared the results between these 2 different software. We have noticed the following different behaviors between SAS and R in data manipulation:

1. Sorting
    1. missing values for numeric values: missing value is sorted before populated values in SAS while missing values is sorted after populated values in R. (Missing values are considered as NA in R)
    2. character values: by default, SAS uses ASCII Sort Order and considers uppercase character smaller than lowercase character: *"A", "B", "a", 'b'.* (We can use sortseq=linguistic to change the feature in SAS); sorting order in R depends on locale setting: locale 'en_US.UTF-8' returns the order *"a","A", "b", "B"* while locale 'C' returns the order *"A", "B", "a", 'b'*. We can use Sys.getlocale('LC_COLLATE') to check the current locale and Sys.setlocale('LC_COLLATE', 'C') to change the locale.
2. Rounding
    1. In R, we can use ROUND or SIGNIF to round. They return an **even** multiple when the input value is halfway between the two nearest multiples of the rounding precision. For example, 1.5 is rounded to 2, 2.5 is rounded to 2, 3.5 is rounded to 4, 4.5 is rounded to 4… (halfway values sometimes round up and sometimes round down).
    2. In SAS, there are 3 rounding functions:

ROUND returns the multiple with the larger absolute value when the input value is approximately halfway between the two nearest multiples of rounding precision. For example, 1.5 is rounded to 2, 2,5 is rounded to 3, 3,5 is rounded to 4, 4.5 is rounded to 5... (halfway values always round up).

ROUNDE returns an even multiple when the input value is approximately halfway between the two nearest multiples of the rounding precision. (halfway values sometimes round up and sometimes round down).

ROUNDZ returns an even multiple when the input value is exactly halfway between the two nearest multiples of the rounding precision. (halfway values sometimes round up and sometimes round down).

(When the rounding unit is less than one and not the reciprocal of an integer, the result that is returned by ROUNDZ might not agree exactly with the result from decimal arithmetic. ROUND and ROUNDE perform extra computations, called fuzzing, to try to make the result agree with decimal arithmetic in the most common situations. ROUNDZ does not fuzz the result.)

3. Precision level in calculation

SAS and R can give different precision level on calculated results:

```
1    data a;
2      point_three=0.3;
3      three_times_point_one=3 * 0.1;
4      difference=point_three - three_times_point_one;
5      put 'The difference is ' difference;
6    run;

The difference is -5.55112E-17
NOTE: The data set WORK.A has 1 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           0.99 seconds
      cpu time            0.12 seconds
```

Result from R:

```
> x<-0.3
> y<-3*0.1
> z<-x-y
> z
[1] -5.551115e-17
>
```

4. Label

In SAS, both dataset and variable have labels. in R, data frame column can have labels. However, data frame does not have label.

5. Length

In SAS, each variable has a predefined length and truncation of character variables can happen due to length. Also before creating XPT files from SAS datasets, we need to redefine the character variable length to reduce the size of the SAS dataset. In R, length of a data frame column is determined by the data. Thus, no truncation will happen and no resize is needed for character columns.

6. Date

SAS uses “1960-01-01” as the reference start date, while R uses “1970-01-01” as the reference start date.

7. Data types

SAS has only 2 data types: numeric and character. R has 5 basic atomic classes: character, doubles, integer, complex, and logical. Special attention need to be paid while manipulating data with r, such as distinguishing between NULL and NA, distinguish between character and logical.

8. Dataset size

R is designed as an in-memory application and loads all data into memory, thus slow while handling large dataset; SAS allocates memory dynamically and handle large dataset better.

CONTACT INFORMATION <HEADING 1>

Your comments and questions are valued and encouraged. Contact the author at:

Yan Qiao
yan.qiao@beigene.com