SESUG Paper 032 - 2025

SAS and R in Action: Comparison of XPT File Creation

Yachen Wang, Chen Ling, AbbVie Inc.

ABSTRACT

SAS is a statistical software used for data analysis and the XPT (SAS Transport File) file format is a standardized way to present data which is widely accepted for clinical trial regulatory submissions. As more and more companies are exploring using R to generate statistical outputs for regulatory submission, there is a growing interest in using R to generate XPT files. However, due to the software setup difference between SAS and R, they handle dataset and variable level attribute differently. In this paper, we introduce the popular ways to assign these attributes in both SAS and R and provide a comprehensive comparison of the process and output. Through practical examples, we will demonstrate how to ensure the exported XPT from SAS and R meet the requirements.

INTRODUCTION

R programming has been increasingly adopted in the pharmaceutical industry in recent years, thanks to its powerful statistical tools, open-source platform, and strong community support. There is a noticeable shift from SAS to R, with many organizations and programmers utilizing R for tasks such as data generation, validation, sharing, and even regulatory submissions. However, the different software setups and environments between SAS and R introduce certain challenges during this transition.

Attributes play a crucial role in data sharing and readability. At the dataset level, SAS and R differ in how they create XPT files: in SAS, the `libname` statement with the `xport` engine is one of the most common methods for generating XPT files, while the `write_xpt` function from the `{haven}` package in R is a popular choice for the same task (Case & Tian, 2021). Besides, these two software employ distinct methods for managing variable attributes, such as LABEL, LENGTH and FORMAT.

In this paper, we will compare how `xport` engine in SAS and `{haven}` package in R generate XPT files at both the dataset and variable levels. Using a sample dummy ADLB dataset, we guide you through each step in detail, highlighting and explaining the differences between the two approaches.

SOURCE DATA

The source data is a dummy ADLB dataset, including USUBJID, PARAMCD, PARAM, ADT, ADTM, ATM, shown in Figure 1.

SUBJID	ADT	ATM	ADTM	PARAM	PARAMCD
1001	2024-04-03	08:18:00	2024-04-03 08:18:00	Calcium (mmol/L)	CCA
1001	2024-04-07	08:01:00	2024-04-07 08:01:00	Calcium (mmol/L)	CCA
1001	2024-04-03	08:18:00	2024-04-03 08:18:00	Chloride (mmol/L)	CCL
1001	2024-04-07	08:01:00	2024-04-07 08:01:00	Chloride (mmol/L)	CCL
1001	2024-04-03	08:18:00	2024-04-03 08:18:00	Potassium (mmol/L)	CK
1001	2024-04-07	08:01:00	2024-04-07 08:01:00	Potassium (mmol/L)	CK
1001	2024-04-03	08:18:00	2024-04-03 08:18:00	Potassium - Hyperkalemia (mmol/L)	CKH
1001	2024-04-07	08:01:00	2024-04-07 08:01:00	Potassium - Hyperkalemia (mmol/L)	CKH
1001	2024-04-03	08:18:00	2024-04-03 08:18:00	Potassium - Hypokalemia (mmol/L)	CKL
1001	2024-04-07	08:01:00	2024-04-07 08:01:00	Potassium - Hypokalemia (mmol/L)	CKL
1001	2024-04-03	08:18:00	2024-04-03 08:18:00	Magnesium (mmol/L)	CMG
1001	2024-04-07	08:01:00	2024-04-07 08:01:00	Magnesium (mmol/L)	CMG
1001	2024-04-03	08:24:00	2024-04-03 08:24:00	pH	UPH
1001	2024-04-07	06:43:00	2024-04-07 06:43:00	pH	UPH
1002	2024-04-03	08:07:00	2024-04-03 08:07:00	Calcium (mmol/L)	CCA

Figure 1. Sample dummy ADLB

DATASET LEVEL

VERSION

There are two versions of the XPT file format currently in use: Version 5 and Version 8. When SAS generates an XPT file using the `xport` engine, it always creates XPT in Version 5 format, regardless of the SAS version being used.

The {haven} package in R supports both Version 5 and Version 8. The `version` option in the `write_xpt` function allows users to select version for the output XPT file. However, it's important to note that SAS 9.3 or earlier cannot read Version 8 XPT files (SAS, SAS Help Center, 2025).

Since the Version 5 XPT file is the official standard compatible with all SAS versions, Pinnacle 21, and submissions, we will focus on discussing the differences in Version 5 XPT files generated by SAS and R in the following sections.

ENCODING

XPT is an older, 8-byte fixed ASCII format used for data interchange; character data is stored as ASCII by default.

R typically uses UTF-8 encoding, but the XPT output is forced to ASCII due to the limitation of the SAS XPT Version 5 format. So non-ASCII characters will be lost in the output process.

DATASET LABEL

A dataset label is mandatory for SDTM and ADaM datasets. It offers a text description that encapsulates the dataset's purpose and content. According to the SDTM/ADaM IG, the dataset label must be less or equal to 40 characters.

In SAS, “LABEL” is a popular method to assign dataset label. When we generate SAS dataset, the dataset is created with this label. A SAS dataset label could be up to 256 characters long, and no errors or warnings if the length is greater than 40 in SAS environment. But anything beyond 40 characters will not be retained during conversion to XPT (Figure 3).

```
*Assign dataset label (<=40);
data adlb1 (label=”Laboratory Test Results Analysis Dataset”);
set adlb;
run;
*Assign dataset label(>40);
data adlb2 (label=”This is Laboratory Test Results Analysis Dataset”);
set adlb;
run;
```

SAS Program 1. assign dataset label

```
P.2   WARNING: Labels exceeding length 40 are not supported by engine XPORT and are being truncated.
P.2   WARNING: Labels exceeding length 40 are not supported by engine XPORT and are being truncated.
```

Figure 3. Truncation when dataset label greater than 40 in SAS

`attr(dataset_name,”label”)` function in R provides capability to check and edit the dataset label. R Program 1 is an example of assigning labels for ADLB dataset (Figure 2). This resulting label will be passed to `write_xpt()` unless `label=NULL` is specified. It's important to note that dataset labels in SAS transport files must be 40 characters or fewer, as per the function setup. A longer dataset label will result in an error in R (Figure 4).

```
#Assign dataset label
attr(adlb,"label")<-“Laboratory Test Results Analysis Dataset”
#Using the assigned label when writing xpt
#Methods 1:
write_xpt(adlb,“Y:/adlb.xpt”)
```

```
#Methods 2:
write_xpt(adlb,"Y:/adlb.xpt", label = attr(data, "label"))
#Not using the assigned label when writing xpt
write_xpt(adlb,"Y:/adlb.xpt",label=NULL)
```

R Program 1. attr(dataset_name,"label") to update dataset label

```
> attr(adlb,"label")<-metadata$label[metadata$dataset =='adlb']
> attr(adlb,"label")
[1] "Laboratory Test Results Analysis Dataset"
```

Figure 2. Output for attr(dataset_name,"label")

```
Error in `write_xpt()`:
! `label` must be 40 characters or fewer.
Run `rlang::last_trace()` to see where the error occurred.
```

Figure 4. Error when dataset label greater than 40 in R

INFORMAT

In SAS, "Informat" column is only useful while inputting datasets. While outputting XPT file, the "Informat" column will be blank unless specified.

Variable	Type	Length	Format	Informat	Label
SUBJID	Character	10			Subject Identifier for the Study
PARAM	Character	20			Parameter
ADT	Numeric	8	DATE9.		Analysis Date
ATM	Numeric	8	TIME5.		Analysis Time
ADTM	Numeric	8	DATETIME20.		Analysis Datetime
PARAMCD	Character	8			Parameter Code

Figure 5. XPT from SAS

{haven} package, on the other hand, takes a more data-preservation-focused approach like other R structures. It automatically assigns a basic "Informat".

Variable	Type	Length	Format	Informat	Label
SUBJID	Character	10			Subject Identifier
ADT	Numeric	8	DATE9.	DATE9.	Analysis Date
ATM	Numeric	8	TIME5.	TIME5.	Analysis Time
ADTM	Numeric	8	DATETIME20.	DATETIME20.	Analysis Datetime
PARAM	Character	20			Parameter
PARAMCD	Character	8			Parameter Code

Figure 6. XPT from R

VARIABLE LEVEL

LABEL

Variable labels in SDTM and ADaM provide clear, consistent descriptions of data variables, facilitating understanding, analysis, and regulatory review by ensuring that all stakeholders can easily interpret.

In SAS, LABEL statement also works for variable labels.

```
*Assign dataset label;
data adlb;
set adlb;
label SUBJID = "Subject Identifier for the Study";
run;
```

SAS Program 2. Assign variable label

Name	Type	Length	Format	Informat	Label
subjid	Character	10			Subject Identifier for the Study
param	Character	20			
ADT	Date	8	DATE9.		
ATM	Time	8	TIME5.		
ADTM	Date	8	DATETIME20.		
PARAMCD	Character	8			

Figure 7. Output for variable label

In {haven} package, `attr()` function is effective for handling attribute tasks. `attr(variable_name,"label")` function can be used to check or to edit a label:

```
> attr(adlb$SUBJID, "label")<-"Subject Identifier for the Study"
> attr(adlb$SUBJID, "label")
[1] "Subject Identifier for the Study"
```

SUBJID Subject Identifier for the Study	ADT	ATM	ADTM	PARAM	PARAMCD

Figure 8. Output for attr(variable_name,"label")

LENGTH

The methods for processing the number of characters in a variable differ between R and SAS.

In SAS, users can explicitly define the number of characters for a variable using “length” statement (Figure 9). If no length is specified, the length of the character is determined as the first assignment or input (Figure 10).

```
*Assign dataset length by using length statement;
data adlb1;
length anl01fl $40;
set adlb;
anl01fl="This is Analysis Flag 1";
run;
*Not assign dataset length;
data adlb1;
set adlb;
anl01fl="This is Analysis Flag 1";
if _n_=15 then anl01fl="This is Analysis Flag 2 for last observation";
run;
```

SAS Program 3. Assign variable length

anl01fl	subjid	param	ADT	ATM	ADTM	PARAMCD
This is Analysis Flag 1	1001	Calcium (mmol/L)	03APR2024	8:18	03APR2024:08:18:00	CCA
This is Analysis Flag 1	1001	Calcium (mmol/L)	07APR2024	8:01	07APR2024:08:01:00	CCA
This is Analysis Flag 1	1001	Chloride (mmol/L)	03APR2024	8:18	03APR2024:08:18:00	CCL
This is Analysis Flag 1	1001	Chloride (mmol/L)	07APR2024	8:01	07APR2024:08:01:00	CCL
This is Analysis Flag 1	1001	Potassium (mmol/L)	03APR2024	8:18	03APR2024:08:18:00	CK
This is Analysis Flag 1	1001	Potassium (mmol/L)	07APR2024	8:01	07APR2024:08:01:00	CK
This is Analysis Flag 1	1001	Potassium - Hyperkal	03APR2024	8:18	03APR2024:08:18:00	CKH
This is Analysis Flag 1	1001	Potassium - Hyperkal	07APR2024	8:01	07APR2024:08:01:00	CKH
This is Analysis Flag 1	1001	Potassium - Hypokale	03APR2024	8:18	03APR2024:08:18:00	CKL
This is Analysis Flag 1	1001	Potassium - Hypokale	07APR2024	8:01	07APR2024:08:01:00	CKL
This is Analysis Flag 1	1001	Magnesium (mmol/L)	03APR2024	8:18	03APR2024:08:18:00	CMG
This is Analysis Flag 1	1001	Magnesium (mmol/L)	07APR2024	8:01	07APR2024:08:01:00	CMG
This is Analysis Flag 1	1001	pH	03APR2024	8:24	03APR2024:08:24:00	UPH
This is Analysis Flag 1	1001	pH	07APR2024	6:43	07APR2024:06:43:00	UPH
This is Analysis Flag 1	1002	Calcium (mmol/L)	03APR2024	8:07	03APR2024:08:07:00	CCA

Figure 9. Output for length statement

subjid	param	ADT	ATM	ADTM	PARAMCD	anl01fl
1001	Calcium (mmol/L)	03APR2024	8:18	03APR2024:08...	CCA	This is Analysis Flag 1
1001	Calcium (mmol/L)	07APR2024	8:01	07APR2024:08...	CCA	This is Analysis Flag 1
1001	Chloride (mmol/L)	03APR2024	8:18	03APR2024:08...	CCL	This is Analysis Flag 1
1001	Chloride (mmol/L)	07APR2024	8:01	07APR2024:08...	CCL	This is Analysis Flag 1
1001	Potassium (mmol/L)	03APR2024	8:18	03APR2024:08...	CK	This is Analysis Flag 1
1001	Potassium (mmol/L)	07APR2024	8:01	07APR2024:08...	CK	This is Analysis Flag 1
1001	Potassium - Hyperkal	03APR2024	8:18	03APR2024:08...	CKH	This is Analysis Flag 1
1001	Potassium - Hyperkal	07APR2024	8:01	07APR2024:08...	CKH	This is Analysis Flag 1
1001	Potassium - Hypokale	03APR2024	8:18	03APR2024:08...	CKL	This is Analysis Flag 1
1001	Potassium - Hypokale	07APR2024	8:01	07APR2024:08...	CKL	This is Analysis Flag 1
1001	Magnesium (mmol/L)	03APR2024	8:18	03APR2024:08...	CMG	This is Analysis Flag 1
1001	Magnesium (mmol/L)	07APR2024	8:01	07APR2024:08...	CMG	This is Analysis Flag 1
1001	pH	03APR2024	8:24	03APR2024:08...	UPH	This is Analysis Flag 1
1001	pH	07APR2024	6:43	07APR2024:06...	UPH	This is Analysis Flag 1
1002	Calcium (mmol/L)	03APR2024	8:07	03APR2024:08...	CCA	This is Analysis Flag 2

Figure 10. Output for not using length statement

When R outputs the XPT file, the length of it will be automatically defined as the largest number of characters presented in the data. `attr(variable_name, "width")` function can facilitate to update length if needed.

Let's take the sample ADLB as an example, if each value we have in PARAMCD is less than 8, we will get less than 8 in `max(nchar())`, which means the length of PARAMCD in the XPT file is also less than 8. Also, the length of PARAM is 33 because the max number of characters of PARAM is 33 in R dataset (Figure 11).

	Variable	Type	Length
1	SUBJID	Character	4
2	ADT	Numeric	8
3	ATM	Numeric	8
4	ADTM	Numeric	8
5	PARAM	Character	33
6	PARAMCD	Character	3

Figure 11. Length of PARAM and PARAMCD in original sample ADLB

`attr (variable_name, "width")` have capability to define the length in the output XPT file and change nothing in R internal dataset. For instance, `attr(adlb1$PARAMCD, "width")<-8` defined the length of PARAMCD as 8 in the sample dummy ADLB (Figure 12).

	Variable	Type	Length
1	SUBJID	Character	4
2	ADT	Numeric	8
3	ATM	Numeric	8
4	ADTM	Numeric	8
5	PARAM	Character	33
6	PARAMCD	Character	8

Figure 12. Assign length to variable

However, if the max number of characters for a variable exceeds the `width` we assign in `attr(variable_name, "width")`, R will give a warning message. Consequently, the variable length in the XPT file will be determined by the maximum number of characters for the variable, according to the R system. The figure 13 below is the example for `attr(adlb1$PARAM, "width")<-30`. Therefore, we need to do data wrangling before using `attr(variable_name, "width")`.

```
Warning message:
Column `PARAM` contains string values longer than user width 30. Width set to 33 to accommodate.
```

Figure 13. Warning from attr(variable_name, "width")

	Variable	Type	Length
1	SUBJID	Character	4
2	ADT	Numeric	8
3	ATM	Numeric	8
4	ADTM	Numeric	8
5	PARAM	Character	33
6	PARAMCD	Character	3

Figure 14. Length of variable is accommodated to 33

As we can see, `attr()` only modifies the length in output .xpt file, and cannot handle those variables whose number of characters exceeds the desired limit. If you want to truncate strings like SAS, R can also achieve this. However, this approach is not recommended, as it may lead to a loss of information, which is not desirable in both SDTM and ADaM datasets. In this section, we will introduce str_sub() function in **{stringr}** package to handle these this issue.

If the max number of characters for a variable is greater than what we desire, function in the **{stringr}** package can be used to substring the value and get the correct number of characters. The max number

of characters for PARAM is 33 in dummy ADLB (Figure 1). By applying `adlb$PARAM<- str_sub (adlb$PARAM, 1,30)`, the number of characters for PARAM will be reduced to 30 (Figure 20).

```
> nchar(adlb$PARAM)
 [1] 16 16 17 17 18 18 30 30 30 30 18 18  2  2 16
> max(nchar(adlb$PARAM))
[1] 30
```

1	SUBJID	Character	4	
2	ADT	Numeric	8	D
3	ATM	Numeric	8	T
4	ADTM	Numeric	8	D
5	PARAM	Character	30	
6	PARAMCD	Character	3	

Figure 15. example for str_sub()

FORMAT

Format is also an essential part in attributes. For variables in SDTM and ADaM dataset, one thing we need to care about is the numeric date and time format.

Usually YYYY-MM-DD or DD-MON-YYYY is used in date, HH:MM or HH:MM:SS in time and DD:MON:YYYY: HH:MM:SS in datetime variable in traditional SAS programming. The corresponding format for them are yymmdd10., date9., time5., time8. and datetime20 (SAS, SAS Help Center, 2017).

In R, date, time and datetime variables are saved in different types: Date, hms, POSIXct type respectively, and their default format are YYYY-MM-DD, HH:MM:SS and YYYY-MM-DD HH:MM:SS. Also, the output XPT file is not well formatted without data processing.

SUBJID Subject Identifier for the Study	ADT Analysis Date	ATM Analysis Time	ADTM Analysis Datetime
1001	2024-04-03	08:18:00	2024-04-03 08:18:00
1001	2024-04-07	08:01:00	2024-04-07 08:01:00
1001	2024-04-03	08:18:00	2024-04-03 08:18:00
1001	2024-04-07	08:01:00	2024-04-07 08:01:00
1001	2024-04-03	08:18:00	2024-04-03 08:18:00
1001	2024-04-07	08:01:00	2024-04-07 08:01:00
1001	2024-04-03	08:18:00	2024-04-03 08:18:00
1001	2024-04-07	08:01:00	2024-04-07 08:01:00
1001	2024-04-03	08:18:00	2024-04-03 08:18:00
1001	2024-04-07	08:01:00	2024-04-07 08:01:00
1001	2024-04-03	08:18:00	2024-04-03 08:18:00
1001	2024-04-07	08:01:00	2024-04-07 08:01:00
1001	2024-04-03	08:24:00	2024-04-03 08:24:00
1001	2024-04-07	06:43:00	2024-04-07 06:43:00
1002	2024-04-03	08:07:00	2024-04-03 08:07:00

```
> class(adlb$ATM)
[1] "hms"      "difftime"
> class(adlb$ADTM)
[1] "POSIXct" "POSIXt"
> class(adlb$ADT)
[1] "Date"
>
```

Figure 16. Default format in R

	Variable	Type	Length	Format	
1	SUBJID	Character	4		
2	ADT	Numeric	8	DATE.	D
3	ATM	Numeric	8	TIME.	T
4	ADTM	Numeric	8	DATETIME.	D
5	PARAM	Character	33		
6	PARAMCD	Character	3		

Figure 17. Default format in .xpt from R

`attr(variable_name, "format.sas")` is useful for assigning formats, the usage is similar with `attr(variable_name, "label")` and `attr(variable_name, "width")` mentioned above. `attr(adlb$ADT,"format.sas")<-"DATE9"` is used to assign SAS "DATE9" format to the ADT variable in

exported XPT file. One important point to note is that the desired format can only be viewed in the XPT file. If we want to customize what displays in R, you need to use Appendix A (UC Berkeley Statistics Department, 2006) to figure out the R format.

	Variable	Type	Length	Format
1	SUBJID	Character	4	
2	ADT	Numeric	8	DATE9.
3	ATM	Numeric	8	TIME5.
4	ADTM	Numeric	8	DATETIME20.
5	ADY	Numeric	8	
6	ADTF	Character	1	
7	PARAM	Character	33	
8	PARAMCD	Character	3	

Figure 18. Output dataset for format

CONCLUSION

In this paper, we present examples using both SAS `xport` as well as `{haven}` in R, comparing them from dataset and variable level. We aim to provide readers with valuable references and information in their SAS to R transition while generating XPT files.

REFERENCES

Case, T., & Tian, Y. (2021). *PharmaSUG 2021.* Retrieved from PharmaSUG 2021: https://pharmasug.org/proceedings/2021/EP/PharmaSUG-2021-EP-057.pdf

SAS. (2017, 3 15). *SAS Help Center*. Retrieved from SAS Help Center: https://documentation.sas.com/doc/en/leforinforref/3.2/p0py1o500c7qsen1hnn2crgejaig.htm

SAS. (2025, 5 9). *SAS Help Center*. Retrieved from SAS Help Center: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/movefile/n1h21k9w9rnhz3n1lrk8s9eib2bt.htm

UC Berkeley Statistics Department. (2006, 2 3). *Dates and Times in R*. Retrieved from Dates and Times in R: https://www.stat.berkeley.edu/~s133/dates.html

APPENDIX

Code	Value
%d	Day of the month (decimal number)
%m	Month (decimal number)
%b	Month (abbreviated)
%B	Month (full name)
%y	Year (2 digit)
%Y	Year (4 digit)
%H	Decimal hours (24 hour)
%M	Decimal Minute
%S	Decimal Second