PharmaSUG 2021 - Paper QT-026

From SAS® Dataset to Structured Document

Hengwei Liu, Daiichi Sankyo, Inc.

ABSTRACT

A structured document is an electronic document whose contents are organized into labeled blocks according to a schema. This is done through a mark-up language such as HTML or XML. If the information to be displayed in the structured document are saved in a SAS dataset, the document can be generated through programming. In this paper we discuss how to use SAS and some other programming languages to do that.

INTRODUCTION

Define.xml is required in FDA submission. It is a structured document. We take a paragraph from the sample define.xml for ADaM in the define.xml v2.1 release package (see reference [1]). This paragraph is the where clause for the analysis result metadata section of the define.xml.

```
<def:WhereClauseDef OID="WC.Table_14-3.01.R.1.ADQSADAS">
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADQSADAS.PARAMCD">
      <CheckValue>ACTOT</CheckValue>
    </RangeCheck>
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADQSADAS.AVISIT">
      <CheckValue>Week 24</CheckValue>
    </RangeCheck>
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADQSADAS.EFFFL">
      <CheckValue>Y</CheckValue>
    </RangeCheck>
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADQSADAS.ANL01FL">
      <CheckValue>Y</CheckValue>
    </RangeCheck>
  </def:WhereClauseDef>
  <def:WhereClauseDef OID="WC.Table_14-3.01.R.2.ADQSADAS">
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADQSADAS.PARAMCD">
      <CheckValue>ACTOT</CheckValue>
    </RangeCheck>
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADQSADAS.AVISIT">
      <CheckValue>Week 24</CheckValue>
    </RangeCheck>
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADQSADAS.EFFFL">
      <CheckValue>Y</CheckValue>
    </RangeCheck>
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADQSADAS.ANL01FL">
```

```
      <CheckValue>Y</CheckValue>
    </RangeCheck>
  </def:WhereClauseDef>
  <def:WhereClauseDef OID="WC.Table_14-5.02.R.1.ADAE">
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADAE.TRTEMFL">
      <CheckValue>Y</CheckValue>
    </RangeCheck>
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADAE.AESER">
      <CheckValue>Y</CheckValue>
    </RangeCheck>
  </def:WhereClauseDef>
  <def:WhereClauseDef OID="WC.Table_14-5.02.R.1.ADSL">
    <RangeCheck Comparator="EQ" SoftHard="Soft" def:ItemOID="IT.ADSL.SAFFL">
      <CheckValue>Y</CheckValue>
    </RangeCheck>
  </def:WhereClauseDef>
```

In this paper we show how to use SAS, R, Python and awk to generate this paragraph. Incidentally we show how R and Python read SAS datasets.

DETAILS

To create the paragraph of text, the first order of business is to set up a SAS dataset that contains the metadata. You can cherry-pick the metadata from the text and ignore the structural elements. Let's call this dataset meta.sas7bdat. Table 1 shows the dataset.

ID	DATASET	VARIABLE	COMPARATOR	CHECKVALUE
Table_14-3.01.R.1.ADQSADAS	ADQSADAS	PARAMCD	EQ	ACTOT
Table_14-3.01.R.1.ADQSADAS	ADQSADAS	AVISIT	EQ	Week 24
Table_14-3.01.R.1.ADQSADAS	ADQSADAS	EFFFL	EQ	Y
Table_14-3.01.R.1.ADQSADAS	ADQSADAS	ANL01FL	EQ	Y
Table_14-3.01.R.2.ADQSADAS	ADQSADAS	PARAMCD	EQ	ACTOT
Table_14-3.01.R.2.ADQSADAS	ADQSADAS	AVISIT	EQ	Week 24
Table_14-3.01.R.2.ADQSADAS	ADQSADAS	EFFFL	EQ	Y
Table_14-3.01.R.2.ADQSADAS	ADQSADAS	ANL01FL	EQ	Y
Table_14-5.02.R.1.ADAE	ADAE	TRTEMFL	EQ	Y
Table_14-5.02.R.1.ADAE	ADAE	AESER	EQ	Y
Table_14-5.02.R.1.ADSL	ADSL	SAFFL	EQ	Y

Table 1. The SAS Dataset Meta

USE SAS TO CREATE THE DOCUMENT

In SAS, data _null_ with put statement can be used to write out the text. Notice the +(-1) that controls the pointer.

```
** wrttxt.sas **;

filename outdir '/xxx/xxx'; * this is the output folder;

data _null_;
set meta;
by id;
file outdir(out.txt);

if first.id then do;
put @1 '<def:WhereClauseDef OID="WC.' id +(-1) '">';
end;

put @1 '<RangeCheck Comparator="' comparator +(-1) '" SoftHard="Soft"
def:ItemOID="IT.' dataset +(-1) '.' variable +(-1) '">';
put @1 '<CheckValue>' checkvalue +(-1) '</CheckValue>';
put @1 '</RangeCheck>';

if last.id then do;
put @1 '</def:WhereClauseDef>';
end;
run;
```

USE R TO CREATE THE DOCUMENT

The R package haven has the function read_sas that can be used to read the SAS dataset. The R function duplicated() is used to achieve what is done by FIRST. and LAST. in SAS. This R program is quite long compared with the SAS program.

```
#wrttxt.R
library(haven)


# Read in the SAS data
met <- read_sas("C:\\xxx\\xxx\\meta.sas7bdat") # folder path
data.frame(met)
id <- met$ID
dataset <- met$DATASET
variable <- met$VARIABLE
comparator <- met$COMPARATOR
checkvalue <- met$CHECKVALUE
```

```
fdot <- !duplicated(met$ID)
ldot <- !duplicated(met$ID,fromLast=TRUE)

# Start writing to an output file
sink('C:\\xxx\\xxx\\output.txt')

for (i in 1:length(fdot))
if (fdot[i]==TRUE & ldot[i]==TRUE) {
  cat(sprintf('<def:WhereClauseDef OID="WC.%s">\n' , id[i] ))
  cat(sprintf('<RangeCheck Comparator="%s" SoftHard="Soft"
def:ItemOID="IT.%s.%s">
  <CheckValue>%s</CheckValue>
  </RangeCheck>\n', comparator[i], dataset[i], variable[i], checkvalue[i]))
  cat(sprintf('</def:WhereClauseDef>\n'))
} else if (fdot[i]==TRUE & ldot[i]==FALSE) {
  cat(sprintf('<def:WhereClauseDef OID="WC.%s">\n' , id[i] ))
  cat(sprintf('<RangeCheck Comparator="%s" SoftHard="Soft"
def:ItemOID="IT.%s.%s">
  <CheckValue>%s</CheckValue>
  </RangeCheck>\n', comparator[i], dataset[i], variable[i], checkvalue[i]))
} else if (fdot[i]==FALSE & ldot[i]==TRUE) {
  cat(sprintf('<RangeCheck Comparator="%s" SoftHard="Soft"
def:ItemOID="IT.%s.%s">
  <CheckValue>%s</CheckValue>
  </RangeCheck>\n', comparator[i], dataset[i], variable[i], checkvalue[i]))
  cat(sprintf('</def:WhereClauseDef>\n'))
} else {
  cat(sprintf('<RangeCheck Comparator="%s" SoftHard="Soft"
def:ItemOID="IT.%s.%s">
  <CheckValue>%s</CheckValue>
  </RangeCheck>\n', comparator[i], dataset[i], variable[i], checkvalue[i]))
}

sink()
```

USE PYTHON TO CREATE THE DOCUMENT

Two Python modules are needed in the program, sys and pandas. The sys module is needed to write out the text. The pandas module has a read_sas function that can read the SAS datasets. The shift() function is used to create two flags that are the same as FIRST. and LAST. in SAS. The function iterrows() is

employed to iterate through each row of the meta dataset.

```
#wrttxt.py
import sys
import pandas as pd

df = pd.read_sas('meta.sas7bdat')
df['FLAG1'] = (df.ID != df.ID.shift()).astype(int)
df['FLAG2'] = (df.ID != df.ID.shift(-1)).astype(int)

sys.stdout = open("OutFile.txt", "w")
for index, row in df.iterrows():
    if row['FLAG1']==1:
      print('<def:WhereClauseDef OID="WC.' + row['ID'].decode() \
              +  '">')
    print('<RangeCheck Comparator="' + row['COMPARATOR'].decode() \
            + '" SoftHard="Soft" def:ItemOID="IT.' + \
          row['DATASET'].decode() + '.' + row['VARIABLE'].decode() \
          + '">')
    print('<CheckValue>' + row['CHECKVALUE'].decode() + '</CheckValue>')
    print('</RangeCheck>')
    if row['FLAG2']==1:
      print('</def:WhereClauseDef>')

sys.stdout.close()
```

USE AWK TO CREATE THE DOCUMENT

The shell script can't directly read the SAS dataset. So the SAS dataset meta is saved as temp.csv. The first step is to add two flags equivalent to the FIRST. and LAST. in SAS to the file temp.csv. We use a script from reference [2] with slight modification.

```
#!/bin/bash
#usage: ./add_flag.sh > meta.csv
file2proc='temp.csv'
tac "${file2proc}" | awk -F ',' ' BEGIN {OFS=FS}
   FNR==NR {
     if(!($1 in last))
         last[$1]=FNR
```

```
     fnr=FNR
     next
   }
 {
     if (FNR==1) {print $0, "FIRST", "LAST";next}
     if(!($1 in first))
       first[$1]=FNR
     print $1, $2, $3, $4, $5, (first[$1]==FNR)?1:0, (last[$1]-1==fnr-
FNR)?1:0

 }
 ' - "${file2proc}" | column  -s ','
```

Output 1 shows the CSV file meta.csv with the two flags FIRST and LAST added.

```
ID,DATASET,VARIABLE,COMPARATOR,CHECKVALUE,FIRST,LAST
Table_14-3.01.R.1.ADQSADAS,ADQSADAS,PARAMCD,EQ,ACTOT,1,0
Table_14-3.01.R.1.ADQSADAS,ADQSADAS,AVISIT,EQ,WEEK 24,0,0
Table_14-3.01.R.1.ADQSADAS,ADQSADAS,EFFFL,EQ,Y,0,0
Table_14-3.01.R.1.ADQSADAS,ADQSADAS,ANL01FL,EQ,Y,0,1
Table_14-3.01.R.2.ADQSADAS,ADQSADAS,PARAMCD,EQ,ACTOT,1,0
Table_14-3.01.R.2.ADQSADAS,ADQSADAS,AVISIT,EQ,WEEK 24,0,0
Table_14-3.01.R.2.ADQSADAS,ADQSADAS,EFFFL,EQ,Y,0,0
Table_14-3.01.R.2.ADQSADAS,ADQSADAS,ANL01FL,EQ,Y,0,1
Table_14-5.02.R.1.ADAE,ADAE,TRTEMFL,EQ,Y,1,0
Table_14-5.02.R.1.ADAE,ADAE,AESER,EQ,Y,0,1
Table_14-5.02.R.1.ADSL,ADSL,SAFFL,EQ,Y,1,1
```

Output 1. CSV File with the Two Flags First and Last Added

Now we are ready to use awk to create the paragraph of text.

```
#Wrttxt.sh

#!/bin/bash

awk -F "," -v var1='<def:WhereClauseDef OID="WC.' \
         -v var2='">' \
      -v var3='<RangeCheck Comparator="' \
      -v var4='" SoftHard="Soft" def:ItemOID="IT.' \
      -v var5='.' \
      -v var6='<CheckValue>' \
      -v var7='</CheckValue>' \
```

```
-v var8='</RangeCheck>' \
-v var9='</def:WhereClauseDef>' \
'(NR>1 && $6==1) {  print var1 $1 var2 "\n" } \
 (NR>1) {print var3 $4 var4 $2 var5 $3 var2 "\n" var6 $5 var7 "\n" var8 "\n"} \
 (NR>1 && $7==1) {print var9 }' meta.csv
```

CONCLUSION

In a structured document, there are a lot of repetitive sentences. With crucial information placed in a SAS dataset, we can use programming to create the structured document. SAS, R and Python can all perform this task. The Linux shell script with awk can also generate the document if the SAS dataset is converted into a CSV file as the input file.

The idea is simple yet powerful. In this paper we showed how to produce the where clause for the analysis result metadata in the define.xml. With all the required metadata set up in some datasets, this idea can be used to generate the define.xml. See reference [3] for an excellent demonstration.

REFERENCES

[1] CDISC Define-XML

https://www.cdisc.org/standards/data-exchange/define-xml

[2] Unix Linux Community

https://community.unix.com/t/how-to-use-awk-to-flag-the-first-and-last-occurence/380616

[3] Jain, Vineet. Robust tools to Create Define.xml v2.0 based submission components for SDTM, ADAM & SEND. PhilaSUG. 2014. Available at http://www.philasug.org/Presentations/201410Fall/Define_Package_Tools_V1_0.pdf

ACKNOWLEDGMENTS

The author thanks the management at Daiichi Sankyo, Inc. for support.

CONTACT INFORMATION

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

Hengwei Liu
Daiichi Sankyo, Inc.
211 Mount Airy Road
Basking Ridge, NJ 07920
Hengwei_liu@yahoo.com

Any brand and product names are trademarks of their respective companies.