Paper SS03

SAS Macro for Converting Variable Attributes from XLSX Specifications to SDTM/ADAM SAS datasets

Jinquan Wu, Pfizer Inc., PA, USA

ABSTRACT

SDTM and ADaM are the standard models required for data submission to regulatory agencies. Their variable attributes, such as variable names, labels, lengths, types, formats are strictly defined in SDTM/ADaM specifications, which usually come in Excel files with XLSX extension. Manually copying the variables and their attributes to SAS program is tedious, time-consuming and error-prone. This paper presents a SAS macro that automatically converts variable attributes from XLSX specifications to SDTM/ADaM SAS datasets that are created in either production or validation process.

Key Words: EXCEL, XLSX Extension, SDTM, ADaM, Specification, Dataset, Converts, Variable Attributes

INTRODUCTION

Both the SDTM and ADaM standards were designed to support submission to regulatory agencies such as the FDA. Data Tabulation Model (SDTM) defines a standard structure for human clinical trial (study) data tabulations and for nonclinical study data tabulations that are to be submitted as part of a product application to a regulatory authority. SDTM dataset specification contains the mappings from the raw data to the SDTM datasets. Analysis Data Model (ADaM) defines datasets and metadata standards that support efficient generation, replication, and review of clinical trial statistical analyses, and traceability among analysis results, analysis data, and data represented in the Study Data Tabulation Model (SDTM). ADaM dataset specification details how to create an ADaM dataset.

Each clinical trial has a SDTM specification and an ADaM specification. These specifications are the blueprints of SDTM/ADaM datasets. The SDTM/ADaM datasets are created strictly based on their respective specifications. However, manually copying SDTM/ADaM dataset names, variables and their attributes to SAS program will be tedious, time-consuming and error-prone. Programmers usually still do manual copy and paste or convert them from Excel SDTM/ADaM specifications to intermediate SAS attribute datasets, and then merge them to SDTM/ADaM datasets.

SAS reads or writes files by using the appropriate engine for that file type. Since SDTM and ADaM specifications usually come in multi-tabbed Excel workbooks with XLSX extension, it requires XLSX engine to read them. This paper will present a macro that automatically converts variable attributes from SDTM/ADaM XLSX specifications to SDTM/ADAM SAS datasets, which requires PC environments with XLSX engine. The macro will greatly enhance programming efficiency and it may be helpful for various level SAS programmers.

SDTM AND ADAM SPECIFICATION

Both SDTM and ADaM specifications are usually multi-tabbed EXCEL workbooks as follows and their structures are similar: Each spreadsheet contains a SDTM domain or an ADaM dataset. The first (or n^{th}) row represents column headers, the other following rows represent distinct variables, and the columns represent metadata attributes, such as domain name or ADaM dataset name, variable name, label, type, length, format, codelist and others. SDTM/ADaM specifications can be summarized as following five types, which are suitable for this macro to read variable attributes from.

Dataset	Variable	Label	Data Type	Length	Significant Digits	Format	Codelist	Origin
DM	STUDYID	Study Identifier	text	200				Protocol
DM	DOMAIN	Domain Abbreviation	text	2			DOMAIN	Assigned
DM	USUBJID	Unique Subject Identifier	text	60				Derived
DM	SUBJID	Subject Identifier for the Study	text	20				CRF
DM	RFSTDTC	Subject Reference Start Date/Time	text	19			ISO 8601	Derived
DM	RFENDTC	Subject Reference End Date/Time	text	19			ISO 8601	Derived
DM	RFXSTDTC	Date/Time of First Study Treatment	text	19			ISO 8601	Derived
DM	RFXENDTC	Date/Time of Last Study Treatment	text	19			ISO 8601	Derived

Study | Datasets | Variables | ValueLevel | WhereClauses | Methods | Comme ...

Snapshot 1. SDTM Specification Sample 1

Name	Label	Sequenc	Length	CodeListRef	Type	Core	Origin	Inform Origin
STUDYID	Study Identifier	1	20		Character	Req	Protocol	
DOMAIN	Domain Abbreviation	2	2	DOMAIN	Character	Req	Assigned	
USUBJID	Unique Subject Identifier	3	60		Character	Req	Derived	
DSSEQ	Sequence Number	4	8		Number	Req	Derived	
DSREFID	Reference ID	5	20		Character	Perm	CRF	DSREFID
DSTERM	Reported Term for the Disposition Event	6	100	DSUNBL	Character	Req	CR, Assigned	DSTERM, DSDECOD
DSDECOD	Standardized Disposition Term	7	50	NCOMPLT	Character	Req	CR, Assigned	DSDECOD
DSCAT	Category for Disposition Event	8	20	DSCAT	Character	Exp	Assigned	
EPOCH	Epoch	9	50	EPOCH	Character	Exp	Derived	
DSDTC	Date/Time of Collection	10	19		Character	Perm	Assigned	
DSSTDTC	Start Date/Time of Disposition Event	11	19		Character	Exp	CRF	DSSTDAT_DTS
DSDY	Study Day of Visit/Collection/Exam	12	8		Number	Perm	Derived	
DSSTDY	Study Day of Start of Disposition Event	13	8		Number	Perm	Derived	
SUPPDS=DSPHASE	Disposition Phase	999	50	EPOCH	Character	Perm	CRF	EPOCH
SUPPDS=DSRCTYP	Re-consent Type(s)	999	25	DSRCTYP	Character	Exp	CRF	DSRCTYP
SUPPDS=DSCSOBT	Consent Was	999	12	DSOBT	Character	Exp	CRF	DSCSOBT

... READ ME | Domain mapping | DS | Change Log

Snapshot 2. SDTM Specification Sample 2

	A	B	C	D	E	F	G	H
1	T-DOMAIN		DS (One record per disposition status or protocol milestone per subject)					
2	T-DESCRIPTION		Disposition					
3	T-MODEL		CaPS SDTM 3.2					
4	T- Class:		EVENTS					
5	PDS Source:		FINAL,STC,ADVERSE, RANDOM					
6	Sort Order / Key:		STUDYID USUBJID DSDECOD DSSTDTC					
7								
8	Variable	Label	Type	Length	Core	CT or Format	Origin	Derivation/Comment
9	STUDYID	Study Identifier	Character	8	Req		Assigned	FINAL.PROTNO,STC.PROTNO,ADVERSE.PROTNO
10	DOMAIN	Domain Abbreviation	Character	2	Req	DOMAIN	Assigned	Default to "DS"
11	USUBJID	Unique Subject Identifier	Character	22	Req		Assigned	FINAL.PID,STC.PID,ADVERSE.PID, RANDOM.PID
12	DSSEQ	Sequence Number	Number	8	Req		Derived	Sequence Number given to ensure uniqueness o
13	DSREFID	Reference ID	Character	5	Perm		RANDOM.PTRAND	equal to RANDOM.PTRAND
14	DSTERM	Reported Term for the Disposition Event	Character	198	Req		FINAL, STC, ADVERSE, RANDOM, Assigned	a. If FINAL.WDRLPVR is not missing or FINSTAT= then set DSTERM = WDRLPVR concatenated wit
15	DSDECOD	Standardized Disposition Term	Character	38	Perm	NCOMPLT.	RANDOM, Assigned	a. If FINAL.WDRLPVR is not missing or FINSTAT=
16	DSCAT	Category for Disposition Event	Character	18	Exp	DSCAT	Assigned	if FINAL.FINSTAT ne . or SBSMNA='NOT APPLICA then DSCAT = 'DISPOSITION EVENT'
17	EPOCH	Epoch	Character	9	Perm	EPOCH	Derived	DSDTC or DSSTDTC is greater than date portion
18	DSDTC	Date/Time of Collection	Character	10	Perm	ISO 8601	STC.COLLEDATE,	dsdtc equal to
19	DSSTDTC	Start Date/Time of Disposition Event	Character	10	Exp	ISO 8601.	FINAL.DIEDDATF, FINAL.WDRLDAT,	a. If DSDECOD = 'DEATH' then DSSTDTC = FINAL.DIEDDATE
20	DSDY	Study Day of Visit/Collection/Exam	Number	8	Perm		Derived	when DSDTC is on or after DM.RFSTDTC, else DSDY = (DSDTC) - (date portion of
21	DSSTDY	Study Day of Start of Disposition Event	Number	8	Perm		Derived	DSSTDY= (DSSTDTC) - (date portion of DM.RFST + 1 when DSSTDTC on or after DM.RFSTDTC,

1056 | DS | SUPPDS

Snapshot 3. SDTM Specification Sample 3

Dataset	Variable	Label	Data Type	Length	Significant Digits	Format	Codelist	Origin
ADSL	STUDYID	Study Identifier	text	20				Predecessor
ADSL	USUBJID	Unique Subject Identifier	text	60				Predecessor
ADSL	SUBJID	Subject Identifier for the Study	text	10				Predecessor
ADSL	SITEID	Study Site Identifier	text	12				Predecessor
ADSL	RANDNO	Randomization Number	text	20				Derived
ADSL	AGE	Age	integer	8				Predecessor
ADSL	AGEU	Age Units	text	6			AGEU	Predecessor
ADSL	AAGE	Analysis Age	integer	8				Derived
ADSL	AAGEU	Analysis Age Unit	text	6			AGEU	Assigned
ADSL	SEX	Sex	text	1			SEX	Predecessor

... Datasets **Variables** ValueLevel WhereClauses Codelists Dictio ...

Snapshot 4. ADaM Specification Sample 1

Var in SDTM (DV or SUPPDV)	Type in SDTM	Core in SDTM	Label in SDTM	CT in SDTM	Var in ADDV	Type in ADDV	Core in ADDV	Label in ADDV	CT in ADAM
STUDYID	Char	Req	Study Identifier		STUDYID	Char	Req	Study Identifier	
USUBJID	Char	Req	Unique Subject		USUBJID	Char	Req	Unique Subject Identifier	
DOMAIN	Char	Req	Domain Abbreviation	DOMAIN	DOMAIN	Char	Req	Domain Abbreviation	
					TRTP	Char	Req	Planned Treatment	
					TRTA	Char	Cond	Actual Treatment	
					TRTPN	Num	Perm	Planned Treatment (N)	
					TRTAN	Num	Perm	Actual Treatment (N)	
					APERSDT	Num		Period Start Date	date9.
DVSEQ	Char	Req	Sequence Number		DVSEQ	Num	Req	Sequence Number	
					DVSPID	Char	Perm	Sponsor-Defined Identifier	
					DVTERM	Char	Req	Protocol Deviation Term	
					DVDECOD	Char	Perm	Protocol Deviation Coded Term	
					DVCAT	Char	Perm	Category for Protocol Deviation	
					DVSCAT	Char	Perm	Subcategory for Protocol Deviation	
					DVSTDTC	Char	Perm	Start Date/Time of Deviation	
					CAPE	Char	Perm	Analysis Population Exclusion	
					ASTDT	Num	Cond	Analysis Start Date	
					ASTDTM	Num	Cond	Analysis Start Date/Time	
					ASTDTF	Char	Cond	Analysis Start Date Imputation Flag	(DATEFL)
					ASTTM	Num	Perm	Analysis Start Time	TIME8.
					ASTTMF	Char	Cond	Analysis Start Time Imputation Flag	
					EPOCH	Char		Epoch	
					ACTSITE	Char		Actual Site of Deviation Occurrence	
					DESGTOR	Char	Perm	Visit Designator	

ADDV Desc **Specifications** ADaM Definition Data & Parameter Files

Snapshot 5. ADaM Specification Sample 2

COMPLETE SAS MACRO

This SAS macro contains seven steps. Key steps are:

Step 2: Read selected spec worksheet with proc import. Usually read entire worksheet. Sometimes only need to read part of worksheet with **RANGE** statement, depending on spec's structure.

Step 3: Convert Specs to an intermediate attribute SAS dataset. Typically, this macro is suitable for above five different types of specifications.

Step 4: Retrieve core variables and their attributes from adsl.

Step 5: Get all attributes ready for use. Ie. put attributes into macro variables and trim variables to minimum length to avoid truncation warning if long length changed to short length

Step 6: Convert the variable attributes to input dataset. For ADaM dataset, ADSL core variables are required to copy to other ADaM datasets. But usually, the core variables are not present in the specification of other ADaM datasets. This macro has option to copy core variables and their attributes from ADSL to other ADaM datasets.

```
%macro util_spec_attrib(specpath=, filename=, worksheet=, range=,  adsl_core_var=, adslpath=, indsn=, outdsn=);

  ** Step 1: Specify Option **;
  option validvarname=UPCASE validmemname=EXTEND nosource  nomlogic nomprint nosymbolgen obs=max;

    ** Step 2: Read Spec Worksheet **;
         %let dbms=%scan(&filename, -1, ".");

     proc import out=attrib0 datafile="&specpath/&filename" dbms=&dbms replace;
            sheet="&worksheet"n;
             getnames=yes;
             %if   &range ^= %then %do;
                    range="&range";
             %end;
       run;

     ** Step 3: Convert Specs to Attrib Dataset **;
        %let dset=open(“attrib0');

         data attrib1;
               attrib dset   length=$25 label=“Dataset Name”
                        vord  length =8.   label=“Variable Order”
                        vnam length=$25 label=“Variable Name”
                         vtyp  length=$25 label=“Variable Type”
                         vlab  length=$40 label=“Variable Label”
                         vlen  length=$25 label=“Variable Length”
                         dfmt  length=$25 label=“Display Format”
                          attr   length =$200 label=“All Attribute”;

            set attrib0;
                 ** Convert Dataset Name **;
                  if %upcase("&worksheet")="VARIABLES" and varnum(&dset, “DATASET”) >0 then do;
                          if upcase(DATASET)=upcase("&outdsn");
                          dset=upcase(DATASET);
                   end;
                   else dset=upcase("&outdsn");

                    ** Convert Variable Name **;
                     if varnum(&dset, “NAME”)>0 then vnam=NAME;
                     else if varnum(&dset, “VARIABLE”)>0 then vnam=VARIABLE;
                     else if varnum(&dset, “VAR_IN_ADAM”)>0 then vnam=VAR_IN_ADAM;
                     else if varnum(&dset, "VAR_IN_&outdsn")>0 then vnam=VAR_IN_&outdsn;

                      if index(upcase(vnam), "=")=0;

                    **Convert Variable Type **;
                     if varnum(&dset, “DATA_TYPE”)>0 then do;
                        if upcase(DATA_TYPE) in ("TEXT", "CHARACTER", "CHAR") then vtyp="CHAR";
                        else if upcase(DATA_TYPE) in ("INTEGER", "NUMBER", "NUM", "FLOAT") then vtyp="NUM";
                      end;
                      else if varnum(&dset, “TYPE”)>0 then do;
                         if upcase(TYPE) in ("TEXT", "CHARACTER", "CHAR") then vtyp="CHAR";
                         else if upcase(TYPE) in ("INTEGER", "NUMBER", "NUM" , "FLOAT") then vtyp="NUM";
                      end;
                      else if varnum(&dset, “TYPE_IN_ADAM”)>0 then do;
                         if upcase(TYPE_IN_ADAM) in ("TEXT", "CHARACTER", "CHAR") then vtyp="CHAR";
                         else if upcase(TYPE_IN_ADAM) in ("INTEGER", "NUMBER", "NUM" , "FLOAT") then vtyp="NUM";
                      end;
                      else if varnum(&dset, "TYPE_IN_&outdsn")>0 then do;
                         if upcase(TYPE_IN_&outdsn) in ("TEXT", "CHARACTER", "CHAR") then vtyp="CHAR";
                         else if upcase(TYPE_IN_&outdsn) in ("INTEGER", "NUMBER", "NUM", "FLOAT") then vtyp="NUM";
                      end;
```

```
          **Convert Variable Label **;
           if varnum(&dset, “LABEL”)>0 then vlab=LABEL;
           else if varnum(&dset, “DESCRIPTION”)>0 then vlab=DESCRIPTION;
           else if varnum(&dset, “LABEL_IN_ADAM”)>0 then vlab=LABEL_IN_ADAM;
           else if varnum(&dset, "LABEL_IN_&outdsn")>0 then vlab=LABEL_IN_&outdsn;

** Convert Variable Format **;
if varnum(&dset, "format")>0 and vtyp="num" and
      prxmatch("m/date|best|$|is8601|iso 8601|yymmdd|time|/i", format)  then dfmt=format;
else if varnum(&dset, "format")=0 and vtyp="num" then do;
       if varnum(&dset, "codelist")>0 and
          prxmatch( "m/date|best|is8601|iso 8601|yymmdd|time|/i", codelist) then dfmt=codelist;
       else if varnum(&dset, 'var5')>0 and
                      prxmatch( "m/date|best|is8601|iso 8601|yymmdd|time|/i", var5)  then dfmt=var5;
       else if vrnum(&dset, 'codelistref')>0 and
            prxmatch("m/date|best|is8601|iso 8601|yymmdd|time|/i", codelistref) then dfmt=codelistref;
       else if varnum(&dset, 'ct_in_adam')>0 and
            prxmatch("m/date|best|is8601|iso 8601|yymmdd|time|/i", ct_in_adam)  then dfmt=ct_in_adam;
       else if varnum(&dset, "ct_in_&outdsn")>0 and
            prxmatch("m/date|best|is8601|iso 8601|yymmdd|time|/i", ct_in_&outdsn) then dfmt=ct_in_&outdsn;
       else if varnum(&dset, "ct_or__format")>0 and
            prxmatch("m/date|best|is8601|iso 8601|yymmdd|time|/i", ct_or__format) then
                                                                  dfmt=compress(ct_or__format, ". ");
       else if varnum(&dset, "ct_or_format")>0 and
             prxmatch("m/date|best|is8601|iso 8601|yymmdd|time|/i", ct_or_format) then
                                                                   dfmt=compress(ct_or_format, ". ");
       else dfmt=" ";
 end;

 **convert variable length **;
  if varnum(&dset, "length")>0 and length^=.  then do;
          if upcase(vtyp)="char" then vlen="$"||strip(put(length, best.));
           else vlen=strip(put(length, best.));
   end;
    else vlen= " " ;

  **remove extra row in spec**;
  if vnam= " " & vlab= " "  & (vlen in (".", " ") or vtyp= " ") then delete;

  ** derived variable attr to hold all attributes **;
  if not missing(vlen) then do;
        if not missing(dfmt) then  attr=strip(vnam)||" length="||strip(vlen)||" label="||strip(vlab)||" format="||strip(dfmt) ;
       else attr=strip(vnam)||" length="||strip(vlen)||" label="||strip(vlab)||"";
  end;
  else  do;
    if not missing(dfmt) then  attr=strip(vnam)||" label="||strip(vlab)||" format="||strip(dfmt) ;
    else attr=strip(vnam)||" label="||strip(vlab)||"";
  end;

   vorder=_N_;

  ** select output dataset  **;
   if dset=upcase("&outdsn");

 run;
```

```
** assign consequent ord # to variables **;
 proc sort data =attrib1 ;
   by dset vorder;
  run;

 data attrib;
    retain dset vord vnam vtyp attr ;
    set attrib1;
    vord=_N_;
  keep dset vord vnam vtyp attr ;
 run;

**step 4: get attributes for ADSL_CORE_VAR from ADSL if necessary **;
 %if "&adsl_core_var" ^= "" %then %do;
     %if %upcase( "&adslpath" ) ^= "" and %upcase( "&adslpath" ) ^= "DATVPROT" %then %do;
           libname adslpath "&adslpath" access=readonly;
           data corevar(keep=&adsl_core_var);
                 retain &adsl_core_var;
                 set adslpath.ADSL;
            run;
%end;
%else %do;
          data corevar(keep=&adsl_core_var);
               retain &adsl_core_var;
               set datvprot.ADSL;
          run;
%end;
%end;

** Step 5: Prepare for Converting Attributes into Input Dataset indsn **;

 ** put vord, vnam, vtyp and attr into macro vars **;
  data _null_;
       set attrib end=final;
       call symputx('n'||strip(vord), vnam);
       call symputx('t'||strip(vord), vtyp);
       call symputx('attr'||strip(vord),attr);
       if final=1 then call symputx('tot', vord);
   run;

 **trim variables to minimum length to avoid truncation warning if long length changed to short length in step 6**;

   %let combvar= ;

    proc sql  noprint;
        %do i=1 %to &tot;
            %if &&t&i = CHAR  %then  %do;
                select  CATS('$', max(length(&&n&i))) into : length from &indsn;
                %let combvar = &combvar _&&n&i &length;
             %end;
          %end;
     quit;

    data trim_&indsn;
          set &indsn;
            length &combvar;
            %do i=1 %to &tot;
                 _&&n&i=&&n&i;
               drop &&n&i;
                rename _&&n&i=&&n&i;
            %end;
      run;
```

```
        *** Step 6: Convert Attributes to Input Dataset indsn ***;

          ** if no length defined in spec, will trim all variables to minimum length **;
            data &outdsn (keep=&n1 -- &&n&tot &adsl_core_var);
                %do i=1 %to &tot;
                 ** convert attributes from spec**;
                    attrib &&attr&i;
                 %end;

                ** get attributes for adsl_core_var if necessary **;
                  %if  "&adsl_core_var"  ^= "" %then %do;
                       if 0 then set corevar;
                  %end;

                set trim_&indsn;
              run;


      ** Step 7: delete intermediate datasets **;
      proc datasets library=work mtype=data nodetails nolist;
      delete attrib: trim_&indsn;
      quit;

%mend;
```

PARAMETER AND APPLICATION

PARAMETER

This macro contains 8 parameters as follows.
specpath: path to the directory where specification files are stored
filename: file name of SDTM/ADAM specification.
worksheet: worksheet name which include all attributes.
range: subsets a worksheet by identifying the rectangular set of cells to import from the
specified worksheet (eg. range=A8:H18. Note no quote on the range. If entire worksheet needs to be read and getname from first row, no need to specify range, just leave it as blank or omit this parameter).
adsl_core_var: list adsl core variables that are directly copied from ADSL to other ADAM datasets.
leave it to blank or omit this parameter if no variable is copied from adsl or they are already listed in spec.
Adslpath: path to the directory where adsl is stored.
set adslpath= blank or datvprot or omit this parameter if adsl is located in data_vai folder of current study.
Indsn: name of input dataset
Outdsn: name of the output dataset (eg. EG, SUPPEG, COEG, ADSL, ADLB, which is listed in the selected worksheet.)

APPLICATION

In this section, three examples are provided to help better understand how to call this macro tool with different situation.

1) For most SDTM/ADAM Spec (eg. Snapshot 1,2 and 4), only five parameters are needed to call the macro for reading entire worksheet and getting column name from first row, eg.

```
%util_spec_attrib(specpath= C:\Bxxx1015\spec,
                  filename= Bxxx1015_dm_mapping_spec.xlsx,
                   worksheet=DM,
                   indsn=dm_final,
                   outdsn=DM);
```

2) For ADAM (eg. Snapshot 5) that need to copy core variables directly from ADSL, two more parameters need to be added (adsl_core_var and adslpath) to call the macro, eg.

```
%util_spec_attrib(specpath= C:\Bxxx1015\spec,
                  filename=Bxxx1015_addv_spec.xlsx,
                  worksheet= Specifications,
                  adsl_core_var= subjid siteid aage aageu sex race ethnic fasfl saffl armcd arm trt01p trt01a trtsdt trtedt,
```

```
adslpath= C:\Bxxx1015\adam,
indsn=addv4,
outdsn=addv);
```

In the CDARS system, ADSL is stored in data_vai folder. So, adslpath can be set as adslpath= , or adslpath=datvprot or omit this parameter.

3) Sometime, spec (eg. Snapshot 3) may contain many more columns or rows than what we need. So, we can use parameter range to read specific columns and rows, eg.

```
%util_spec_attrib(specpath= C:\Axxx1056\spec,
                  filename=Axxx1056_pds_ds.xlsx,
                  worksheet=DS,
                  range=A8:H21,
                  indsn=DS8,
                  outdsn=DS);
```

CONCLUSION

This SAS macro provides an automatic method to convert variable attributes directly and quickly from SDTM/ADaM XLSX specifications to SDTM/ADaM datasets. This method significantly improves programming efficiency and accuracy when producing or validating SDTM and ADaM datasets. This macro is portable with minimal programmer intervention and so helpful for various level SAS programmers.

REFERENCE

1. DelGobbo, Vince. 2019. “Integrating SAS® and Microsoft Excel: Exploring the Many Available to You”. Proceedings of PharmaSUG 2019. Philadelphia, PA. Available at https://www.lexjansen.com/pharmasug/2019/HT/PharmaSUG-2019-HT-067.pdf
2. Zhong, Wayne. 2012. “Efficiently Trim Character Variable Lengths to Fit Data, Reduce Dataset Size”. Proceedings of PharmaSUG 2012. San Francisco, CA. Available at https://www.pharmasug.org/proceedings/2012/CC/PharmaSUG-2012-CC17.pdf

ACKNOWLEDGMENTS

Thank Gabriela Piasecki, Gori Maya Gurung and Ajay Krishna Shrestha for testing this macro.

Thank LiFeng Good for reviewing the paper and providing additional insight.

CONTACT INFORMATION

Jinquan Wu
Pfizer, Inc.
500 Arcola Rd, Collegeville, PA 19426
jinquan.wu6@gmail.com
https://www.pfizer.com/