PharmaSUG 2014 – Paper AD26

Validating Analysis Data Set without Double Programming - An Alternative Way to Validate the Analysis Data Set

Linfeng Xu, Novartis, East Hanover, NJ
Christina Scienski, Novartis, East Hanover, NJ

ABSTRACT

This paper will demonstrate an alternative way to validate the analysis data set without double programming. The common practice for the most critical level validation of an analysis data set in pharmaceutical industry is double programming, that is, the source programmer generates an analysis data set (i.e. production data set) and another programmer (Reviewer) uses same specifications to create a QC data set. Reviewer then conducts PROC COMPARE to see whether or not there is any discrepancy between the two data sets. This paper will introduce the ALTVAL macro as an alternative to double programming in validating an analysis data set. The ALTVAL macro consists of two parts: 1) compare common variables between raw and analysis data set and use PROC COMPARE to check for discrepancies. (common variable check) 2) using unique merging variable(s), merge the raw and analysis data set to create a new data set, conduct a cross-frequency check between input variables and derived variables (cross check); 3) Based upon the step two created big combined data set, reviewer can write simple codes to report the cases which could not meet the variable derivation rules in the specification since we have input variables and derived variables in same big combined data set (the logic check of variable specification). This paper will discuss the conceptual design, macro parameters, and prerequisites using this new approach. It will also discuss what types of analysis data sets are suitable for this new validation approach.

BACKGROUND

Based on the importance of a data set, complexity of derived variables in an analysis data set, availability of resource, including manpower and time to conduct the validation, we usually assign varying levels of validation. For the most critical validation, double programming is the current standard practice for verifying whether or not the derived analysis data set is accurate and correct based on specification. A reviewer may also conduct code review, log check, and data set check in addition to the double programming. Double programming is very time-consuming and thus a very expensive way for validation. This paper describes an alternative approach in validating an analysis data set for the most critical level of validation, which can potentially save time and money substantially. One caveat is that this alternative approach will not replace the steps that involve code review, log check, and data set check.

INTRODUCTION

CONCEPTUAL DESIGN

When comparing raw data sets and a derived analysis data set, one needs to be aware that there are two types of variables in terms of source in the derived analysis data set. One type of variables is “common variables” that have the same variable names and data attributes as those in the raw data sets. The other type of variables is “derived variables” with different variable names and data attributes, which are derived from source variable(s) in the source data sets. For common variables, if a PROC COMPARE ID variable can be found, one can run PROC COMPARE using an ID statement. This first step is the same as that used in a double programming approach. For derived variables, however, since a validation data set will not be created, one cannot use PROC COMPARE to validate the derived variables. In the ALTVAL macro proposed by this paper, there is an alternative way to validate derived variables. First, one needs to identify or create unique merging variable(s) and use this/these variable(s) as the key to merge the raw data sets and derived analysis data set into one data set. During the merging process, we suggest

keep the common variables from the derived analysis data set since they are the same as those in the raw data sets. Using this merged dataset, a cross-frequency check (i.e. “cross check”) is run to compare the derived variables from the derived analysis data set with those original variables from the raw data set. A subset of observations that could not meet the derivation rules will be reported as discrepancies (“the logic check of variable specification”). For the cross check, reviewer will manually review the outputs against derivation rules defined in analysis derivation specification. For example, one can check the age variable in ADSL against the birth date variable in raw DM data set and first dose date variable in ADSL by running cross frequency age*birthdtc*trt01sdt if the first dose date is used as a reference variable to impute age. For the logic check of variable specification, one can check whether or not there are any cases listed in output that did not meet the derivation rule for imputing the age variable: age ^= (trt01sdt-input(birthdtc,yymmdd8.)/365.25). (Note: in this case trt01sdt is a SAS date variable, and birthdtc is a character variable like 19650326). To conclude, this macro assists the validator in understanding the input data that is needed to derive a variable and more easily pick up discrepancies that could happen upon derivation for the analysis dataset.

ALTVAL MACRO

```
/************************************************************************
** FILENAME : ALTVAL.SAS
** DEVELOPER: LINFENG XU
** DATE:      07NOV2013
** PROJECT
** DESCRIPTION: VALIDATION FOR TWO DATA SETS
** MACROS USED:
** INPUT:
** OUTPUT:
** MODIFICATION HISTORY:
** MODIFIED BY:                 DATE:
** 07NOV2013 LINFENG XU CREATION
************************************************************************/;

%MACRO ALTVAL(SILIB=DATA_S,SDLIB=DATA_A,SIDB=,SDDB=,IDVARS=,FRQVAR=N,
            SHOWCFRQ=N,SHOWIFRQ=N,SHOWDFRQ=N,OUTDB=N) ;
/** ASSUMPTIONS:
**           SILIB=LIBNAME FOR 1ST INPUT DATA SET
**           SDLIB=LIBNAME FOR 2ND INPUT DATA SET
**           SIDB=1ST DATA SET
**           SDDB=2ND DATA SET
**           IDVARS=ID VARIABLE NAME LIST FOR PROC COMPARE
**           FRQVAR=FREQUENCY VARIABLE PAIR LIST-ONE FROM 1ST DATA SET AND
**                 ONE FROM 2ND DATA SET AND ASSIGNED AS %STR(VAR1*VAR2)
**           OUTDB=NAME OF THE BIG COMBINED DATA SET
**           SHOWCFRQ=Y -PROVIDE FREQUENCY FOR COMMON VARIABLES EXIST IN BOTH
**                      DATA SETS
**           SHOWIFRQ=Y -PROVIDE FREQUENCY FOR VARIABLES ONLY EXISTED IN SIDB
**           SHOWDFRQ=Y -PROVIDE FREQUENCY FOR VARIABLES ONLY EXISTED IN SDDB
**/

OPTIONS NOFMTERR MPRINT NOMLOGIC ;

PROC CONTENTS DATA=&SDLIB..&SDDB NOPRINT OUT=C&SDDB(KEEP=NAME);
RUN;

PROC SORT DATA=C&SDDB;
```

```
BY NAME;
RUN;

PROC CONTENTS DATA=&SILIB..&SIDB NOPRINT OUT=D&SIDB(KEEP=NAME);
RUN;

PROC SORT DATA=D&SIDB;
BY NAME;
RUN;

DATA SAME ONLYDRV ONLYRAW;
MERGE C&SDDB(IN=A KEEP=NAME) D&SIDB(IN=B KEEP=NAME);
BY NAME;
IF A AND B THEN OUTPUT SAME;
IF A AND NOT B THEN OUTPUT ONLYDRV;
IF NOT A AND B THEN OUTPUT ONLYRAW;
RUN;

PROC PRINT DATA=SAME;
VAR NAME;
TITLE "COMMON VARIABLES IN BOTH &SIDB AND &SDDB";
RUN;

PROC PRINT DATA=ONLYDRV;
VAR NAME;
TITLE "VARIABLES IN ONLYDRV";
RUN;

PROC PRINT DATA=ONLYRAW;
VAR NAME;
TITLE "VARIABLES IN ONLYRAW";
RUN;

PROC SQL NOPRINT;
SELECT DISTINCT NAME INTO: CNAME SEPARATED BY " " FROM SAME(WHERE=(NAME NOT IN
("USUBJID","VISITNUM","VISITNU","VISIT")))
;
SELECT DISTINCT NAME INTO: SINAME SEPARATED BY " " FROM ONLYRAW;
;
SELECT DISTINCT NAME INTO: SDNAME SEPARATED BY " " FROM ONLYDRV;
;
QUIT;

%PUT &CNAME;

%IF &SHOWCFRQ=Y %THEN %DO;
PROC FREQ DATA=&SDLIB..&SDDB ;
TABLE &CNAME /LIST MISSING;
TITLE "FREQ OF COMMON VARIABLES IN &SDDB";
RUN;
%END;
```

```
%IF &SHOWIFRQ=Y %THEN %DO;
PROC FREQ DATA=&SILIB..&SIDB ;
TABLE &SINAME /LIST MISSING;
TITLE "FREQ OF  VARIABLES ONLY IN &SIDB";
RUN;
%END;

%IF &SHOWDFRQ=Y %THEN %DO;
PROC FREQ DATA=&SDLIB..&SDDB;
TABLE &SDNAME/LIST MISSING;
TITLE "FREQ OF VARIABLES ONLY IN &SDDB";
RUN;
%END;

PROC SORT DATA=&SDLIB..&SDDB OUT=S&SDDB;
BY &IDVARS;
RUN;

PROC SORT DATA=&SILIB..&SIDB OUT=F&SIDB;
BY &IDVARS;
RUN;

PROC COMPARE DATA=S&SDDB COMP=F&SIDB LISTALL MAXPRINT=(20,32760)  ;
VAR &CNAME;
ID &IDVARS;
TITLE "COMPARE IN=S&SDDB WITH COMP=F&SIDB COMMON VARIABLES";
RUN;

%IF &FRQVAR^=N %THEN %DO;

DATA DIFF;
MERGE F&SIDB(IN=A) S&SDDB(IN=B KEEP= &IDVARS &SDNAME );
BY &IDVARS;
IF A OR B;
RUN;

PROC FREQ DATA=DIFF;
TABLE &FRQVAR/LIST MISSING;
TITLE "CROSS CHECK PAIR VARIABLES";
RUN;
%END;

%IF &OUTDB^=N %THEN %DO;
*NOTE: &OUTDB ASSIGNED CAN INCLUDE LIBNAME AND DATA SET NAME;
DATA &OUTDB;
 SET DIFF;
RUN;
%END;
%MEND ALTVAL;
```

PREREQUISITES

In order to use ALTVAL macro, it is very important to identify unique merging variable(s) as merging variable(s). This/these unique merging variable(s) is/are used in both steps of the validation: first it is used as ID variable (s) in PROC COMPARE for common variables, second it is used to merge the raw data set(s) with the derived analysis data. If such variable (s) do not exist in both raw data set and derived analysis data set, one needs to create one. A quick and simple way to confirm unique merging variable(s) is to run a PROC SORT by NODUPKEY to see if any observations were removed from the testing data set. If yes, additional variable(s) need to be added to form unique merging variable(s). For example, in the ADSL data set, unique merging variable(s) is USUBJID. In the ADLB data set, however, variables that form the unique merging variable(s) include USUJIBD, PARAMCD (parameter code), and ADT (analysis date). Since both variables PARAMCD and ADT are not in the raw data set LB, they need to be created in the raw data set. The variable PARAMCD can be created by using LBTESTCD (lab test code) and LBSTRESU (lab test result unit), and the variable ADT can be created from datepart(LBDTC) (lab date).

MACRO PARAMETERS AND THEIR FUNCTION

In order to increase the flexibility of using the ALTVAL macro in different situations, the input data set LIBNAME acronym (SILIB), derived analysis data set LIBNAME acronym (SDLIB), input data set name (SIDB), analysis data set name (SDDB) are defined as macro parameters. This gives the user ability to assign acronyms for the raw data sets and derived analysis data set. The user can define acronyms for the data set library by using LIBNAME statements for SILIB and SDLIB. WORK can also be used as SILIB and SDLIB if there is any pre-processing prior to macro call.
IDVARS is used to identify the unique merging variable(s).
FRQVAR can be assigned as cross frequency variable format, such as var1*var2. It can have more than two variables. It also can have more than one check, such as var1*var2 var3*var4*var5.
OUTDB can be used to output the big combined data set to a permanent data set. It can be assigned with the combined LIBNAME acronym and data set name, such as data_a.adsl_qc.
User can assign a “Y” value to SHOWCFRQ and this will generate PROC FREQ output for all the common variables from the derived analysis data set. The default value is `N`.
User can assign a “Y” value to SHOWIFRQ and this will generate PROC FREQ output for all the unique variables that only exist in the input data set. The default value is `N`.
User can assign a “Y” value to SHOWDFRQ and this will generate PROC FREQ output for all the unique variables that only exist in the derived analysis data set. The default value is `N`.

After the ALTVAL macro is successfully executed, the following outputs will be generated: 1) list of variables that only exist in the input data set; 2) list of variables that only exist in the derived analysis data set; 3) list of variables that exist in both input data set and derived analysis data set; 4) PROC FREQ output for all the common variables in the derived data set, if SHOWCFRQ=Y;5) PROC FREQ output for all the unique variables in the input data set, if SHOWIFRQ=Y;6) PROC FREQ output for all the unique variables in the derived analysis data set, if SHOWDFRQ=Y; and 7) a combined data set (work.diff) if &OUTDB is not defined or a permanent data set in &OUTDB (User can use either DIFF or &OUTDB to write user defined codes to conduct the logic check of variable specification).

SAMPLE MACRO CALL AND THE OUTPUTS OF ALTVAL MACRO

```
*CREATE SAMPLE VS DATA SET WITH ONE PATIENT AND ITS VISITS;
DATA VS;
INFILE DATALINES DSD MISSOVER;
ATTRIB STUDYID FORMAT=$7. LABEL="STUDY IDENTIFIER";
ATTRIB DOMAIN FORMAT=$2. LABEL="DOMAIN ABBREVIATION";
ATTRIB USUBJID FORMAT=$15. LABEL="UNIQUE SUBJECT IDENTIFIER";
ATTRIB VSSEQ FORMAT=8.0 LABLE="SEQUENCE NUMBER";
ATTRIB VISITNUM FORMAT=8.0 LABEL="VISIT NUMBER";
```

```
ATTRIB VSORRES FORMAT=8.3 LABEL="RESULT OR FINDING IN ORIGINAL UNITS";
ATTRIB VSSTRESN FORMAT=8.3 LABEL="NUMERIC RESULT/FINDING IN STANDARD UNITS";
ATTRIB VISIT FORMAT=$9. LABEL="VISIT NAME";
ATTRIB VSTESTCD FORMAT=$5. LABEL="VISIT SIGNS TEST SHORT NAME";
ATTRIB VSTEST FORMAT=$10. LABEL="VITAL SIGNS TEST NAME";
ATTRIB VSPOS FORMAT=$8. LABEL="VITAL SIGNS POSITION OF SUBJECT";
ATTRIB VSCAT FORMAT=$7. LABEL="CATEGORY FOR VITAL SIGNS";
ATTRIB VSORRESU FORMAT=$9. LABEL="ORIGINAL UNITS";
ATTRIB VSSTRESU FORMAT=$9. LABEL="STANDARD UNITS";
ATTRIB VSDTC FORMAT=$10. LABEL="DATE/TIME OF MEASUREMENTS";


INPUT STUDYID $ DOMAIN $ USUBJID $ VSSEQ  VISIT $ VISITNUM   VSTESTCD $  VSTEST
$ VSCAT $
VSPOS $ VSORRES   VSORRESU $ VSSTRESN    VSSTRESU $ VSDTC;
DATALINES;
ABC2201,VS,ABC2201_1001001,21,SCREENING,1.000,PULSE,PULSE
RATE,GENERAL,STANDING,70.00,BEATS/MIN,70,BEATS/MIN,2012-08-21
ABC2201,VS,ABC2201_1001001,22,BASELINE,101.000,PULSE,PULSE
RATE,GENERAL,STANDING,80.00,BEATS/MIN,80,BEATS/MIN,2012-08-28
ABC2201,VS,ABC2201_1001001,23,WEEK1,102.000,PULSE,PULSE
RATE,GENERAL,STANDING,78.00,BEATS/MIN,78,BEATS/MIN,2012-09-04
ABC2201,VS,ABC2201_1001001,24,WEEK2,103.000,PULSE,PULSE
RATE,GENERAL,STANDING,85.00,BEATS/MIN,85,BEATS/MIN,2012-09-11
ABC2201,VS,ABC2201_1001001,25,WEEK4,104.000,PULSE,PULSE
RATE,GENERAL,STANDING,86.00,BEATS/MIN,86,BEATS/MIN,2012-09-28
ABC2201,VS,ABC2201_1001001,26,WEEK8,105.000,PULSE,PULSE
RATE,GENERAL,STANDING,77.00,BEATS/MIN,77,BEATS/MIN,2012-10-29
ABC2201,VS,ABC2201_1001001,27,WEEK12,106.000,PULSE,PULSE
RATE,GENERAL,STANDING,97.00,BEATS/MIN,97,BEATS/MIN,2012-11-23
;
RUN;

PROC PRINT DATA=VS;
TITLE "VS";
RUN;

*CREATE SAMPLE ADSL FOR 1 PATIENT;
DATA ADSL0;
INFILE DATALINES DSD MISSOVER;
ATTRIB STUDYID FORMAT=$7. LABEL="STUDY IDENTIFIER";
ATTRIB DOMAIN FORMAT=$4. LABEL="DOMAIN ABBREVIATION";
ATTRIB USUBJID FORMAT=$15. LABEL="UNIQUE SUBJECT IDENTIFIER";
ATTRIB SEX FORMAT=$1. LABEL="SEX";
ATTRIB RACE FORMAT=$9. LABEL="RACE";
ATTRIB AGE FORMAT=8.0 LABEL="AGE";
ATTRIB FASFL FORMAT=$1. LABEL="FULL ANALYSIS SET POPULATION FLAG";
ATTRIB SAFFL FORMAT=$1. LABEL="SAFETY POPULATION FLAG";
ATTRIB TRTSDT FORMAT=$10. LABEL="DATE OF FIRST EXPOSURE TO TREATMENT";
ATTRIB TRTEDT FORMAT=$10. LABEL="DATE OF LAST EXPOSURE TO TREATMENT";
INPUT  STUDYID $ DOMAIN $ USUBJID $ SEX $ RACE $ FASFL $ SAFFL $  TRTSDT
$ TRTEDT $ AGE;
DATALINES;
ABC2201,ADSL,ABC2201_1001001,F,CAUCASIAN,Y,Y,2012-08-28,2012-12-11,63
;
RUN;

DATA ADSL;
SET ADSL0;
ATTRIB TRT01SDT FORMAT=IS8601DA. LABEL="DATE OF FIRST EXPOSURE TO TREATMENT";
```

```
ATTRIB TRT01EDT FORMAT=IS8601DA.  LABEL="DATE OF LAST EXPOSURE TO TREATMENT";
TRT01SDT=INPUT(TRTSDT,YYMMDD10.);
TRT01EDT=INPUT(TRTEDT,YYMMDD10.);
RUN;

*CREATE SAMPLE ADVS DATA SET WITH SERVERAL DERIVED VARIABLES;
PROC SQL;
CREATE TABLE ADVS1 AS SELECT DISTINCT
A.*
,B.TRT01SDT
,B.TRT01EDT
FROM VS AS A LEFT JOIN ADSL AS B
ON A.USUBJID=B.USUBJID
;
QUIT;

DATA ADVS2(DROP=VSDTC VSTESTCD VSTEST VSORRES VSORRESU VSSTRESN VSSTRESU VSPOS
VSCAT DOMAIN);
SET ADVS1;

ATTRIB AVISITN LABEL="ANALYSIS VISIT (N)";
IF VISITNUM^=. THEN AVISITN=VISITNUM;

ATTRIB AVISIT FORMAT=$20. LABEL="ANALYSIS VISIT";
IF VISIT^="" THEN AVISIT=VISIT;


ATTRIB ADT  FORMAT=IS8601DA. LABEL="ANALYSIS DATE";
IF VSDTC^="" THEN ADT=INPUT(VSDTC,YYMMDD10.);

ATTRIB ADY  LABEL="ANALYSIS RELATIVE DAY";
IF INPUT(VSDTC,YYMMDD10.)>=TRT01SDT THEN ADY=INPUT(VSDTC,YYMMDD10.)-TRT01SDT+1;
ELSE ADY=INPUT(VSDTC,YYMMDD10.)-TRT01SDT;

ATTRIB ANL01FL FORMAT=$1. LABEL="ANALYSIS RECORD FLAG 01";
IF      NMISS(ADT,TRT01SDT,TRT01EDT)=0 AND TRT01SDT<=ADT<=TRT01EDT THEN
ANL01FL="Y";
ELSE IF NMISS(ADT,TRT01SDT)=0 AND TRT01EDT=. AND TRT01SDT<=ADT THEN ANL01FL="Y";
ELSE IF NMISS(ADT,TRT01SDT)=0   AND ADT<TRT01SDT  THEN ANL01FL="N";

ATTRIB ANL01FN  LABEL="ANALYSIS RECORD FLAG 01 (N)";
IF ANL01FL="Y" THEN ANL01FN=1;
ELSE ANL01FN=0;

ATTRIB ANL01DSC FORMAT=$45. LABEL="ANALYSIS RECORD FLAG 01 DESCRIPTION";
ANL01DSC="ANALYSIS RECORD FLAG FOR DOUBLE-BLIND PERIOD - VS";


ATTRIB AVAL FORMAT=BEST12. LABEL="ANALYSIS VALUE";
IF VSSTRESN^=. THEN AVAL=VSSTRESN;

ATTRIB PARAMCD FORMAT=$20. LABEL="PARAMETER CODE";
ATTRIB PARAM FORMAT=$40. LABLE="PARAMETER";
ATTRIB PARAMN LABEL="PARAMETER(N)";

IF VSTESTCD="PULSE" AND VSPOS="STANDING" THEN DO;
PARAMCD="STDPULSE";
PARAM="STANDING PULSE RATE ("!!STRIP(VSSTRESU)!!")";
PARAMN=1;
END;
```

```
RUN;

PROC SORT DATA=ADVS2(WHERE=(.<ADT<=TRT01SDT AND AVAL^=.)) OUT=BASE;
BY USUBJID ADT;
RUN;

DATA BASE;
SET BASE;
BY USUBJID ADT;
IF LAST.USUBJID THEN OUTPUT;
RUN;

PROC SQL;
CREATE TABLE ADVS3 AS SELECT DISTINCT
A.*
,B.AVAL AS BASE FORMAT=BEST12. LABEL="BASELINE VALUE"
,CASE
WHEN NMISS(B.AVAL,A.AVAL)=0 THEN (A.AVAL-B.AVAL)
ELSE .
END AS CHG FORMAT=8.2 LABEL="CHANGE FROM BASELINE"
,CASE
WHEN  NMISS(B.AVAL,A.AVAL)=0 THEN 100*(A.AVAL-B.AVAL)/B.AVAL
ELSE .
END AS PCHG FORMAT=8.2 LABEL="PERCENTAGE CHANGE FROM BASELINE"
,CASE
WHEN A.AVISITN=B.AVISITN THEN "Y"
ELSE "N"
END AS ABLFL FORMAT=$1. LABEL="BASELINE RECORD FLAG"

FROM ADVS2 AS A LEFT JOIN BASE AS B
ON A.USUBJID=B.USUBJID
ORDER BY A.USUBJID,A.PARAMN,A.ADT,A.AVISITN
;
QUIT;

DATA ADVS;
SET ADVS3;
ATTRIB ABLFL FORMAT=$1. LABEL="BASELINE RECORD FLAG";
IF  ABLFL="Y" THEN ABLFN=1;
ELSE ABLFN=0;

ATTRIB CRIT1 FORMAT=$45. LABEL="ANALYSIS CRITERION 1";
CRIT1=">=120 AND >=15 INCREASE FROM BASELINE";
ATTRIB CRIT2 FORMAT=$45. LABEL="ANALYSIS CRITERION 2";
CRIT2="<=50 AND >=15 DECREASE FROM BASELINE";

ATTRIB CRIT1FL FORMAT=$1. LABEL="CRITERION 1 EVALUATION RESULT FLAG";
ATTRIB CRIT1FN LABEL="CRITERION 1 EVALUATION RESULT FLAG (N)";
ATTRIB CRIT2FL FORMAT=$1. LABEL="CRITERION 2 EVALUATION RESULT FLAG";
ATTRIB CRIT2FN LABEL="CRITERION 2 EVALUATION RESULT FLAG (N)";

IF AVAL>=120 AND CHG>=15 THEN DO;
CRIT1FL="Y";
CRIT1FN=1;
END;
ELSE DO;
CRIT1FL="N";
CRIT1FN=0;
END;
IF AVAL<=50 AND CHG<=-15 THEN DO;
```

```
CRIT2FL="Y";
CRIT2FN=1;
END;
ELSE DO;
CRIT2FL="N";
CRIT2FN=0;
END;
RUN;
```

Output showing the raw data VS.

VS

Obs	STUDYID	DOMAIN	USUBJID	VSSEQ	VISITNUM	VSORRES	VSSTRESN	VISIT
1	ABC2201	VS	ABC2201_1001001	21	1	70.000	70.000	SCREENING
2	ABC2201	VS	ABC2201_1001001	22	101	80.000	80.000	BASELINE
3	ABC2201	VS	ABC2201_1001001	23	102	78.000	78.000	WEEK1
4	ABC2201	VS	ABC2201_1001001	24	103	85.000	85.000	WEEK2
5	ABC2201	VS	ABC2201_1001001	25	104	86.000	86.000	WEEK4
6	ABC2201	VS	ABC2201_1001001	26	105	77.000	77.000	WEEK8
7	ABC2201	VS	ABC2201_1001001	27	106	97.000	97.000	WEEK12

Obs	VSTESTCD	VSTEST	VSPOS	VSCAT	VSORRESU	VSSTRESU	VSDTC
1	PULSE	PULSE RATE	STANDING	GENERAL	BEATS/MIN	BEATS/MIN	2012-08-21
2	PULSE	PULSE RATE	STANDING	GENERAL	BEATS/MIN	BEATS/MIN	2012-08-28
3	PULSE	PULSE RATE	STANDING	GENERAL	BEATS/MIN	BEATS/MIN	2012-09-04
4	PULSE	PULSE RATE	STANDING	GENERAL	BEATS/MIN	BEATS/MIN	2012-09-11
5	PULSE	PULSE RATE	STANDING	GENERAL	BEATS/MIN	BEATS/MIN	2012-09-28
6	PULSE	PULSE RATE	STANDING	GENERAL	BEATS/MIN	BEATS/MIN	2012-10-29
7	PULSE	PULSE RATE	STANDING	GENERAL	BEATS/MIN	BEATS/MIN	2012-11-23

Output 1. RAW Data VS

Output showing the derived analysis data set - ADVS.

ADVS

Obs	STUDYID	USUBJID	VSSEQ	VISITNUM	VISIT	TRT01SDT	TRT01EDT	AVISITN	AVISIT
1	ABC2201	ABC2201_1001001	21	1	SCREENING	2012-08-28	2012-12-11	1	SCREENING
2	ABC2201	ABC2201_1001001	22	101	BASELINE	2012-08-28	2012-12-11	101	BASELINE
3	ABC2201	ABC2201_1001001	23	102	WEEK1	2012-08-28	2012-12-11	102	WEEK1
4	ABC2201	ABC2201_1001001	24	103	WEEK2	2012-08-28	2012-12-11	103	WEEK2
5	ABC2201	ABC2201_1001001	25	104	WEEK4	2012-08-28	2012-12-11	104	WEEK4

Obs	ADT	ADY	ANL01FL	ANL01FN	ANL01DSC	AVAL
1	2012-08-21	-7	N	0	Analysis record flag for Double-Blind Period	70
2	2012-08-28	1	Y	1	Analysis record flag for Double-Blind Period	80
3	2012-09-04	8	Y	1	Analysis record flag for Double-Blind Period	78
4	2012-09-11	15	Y	1	Analysis record flag for Double-Blind Period	85
5	2012-09-28	32	Y	1	Analysis record flag for Double-Blind Period	86

Obs	PARAMCD	PARAM	PARAMN	BASE	CHG
1	STDPULSE	Standing Pulse Rate (BEATS/MIN)	1	80	-10.00
2	STDPULSE	Standing Pulse Rate (BEATS/MIN)	1	80	0.00
3	STDPULSE	Standing Pulse Rate (BEATS/MIN)	1	80	-2.00
4	STDPULSE	Standing Pulse Rate (BEATS/MIN)	1	80	5.00
5	STDPULSE	Standing Pulse Rate (BEATS/MIN)	1	80	6.00

Obs	PCHG	ABLFL	ABLFN	CRIT1	CRIT2
1	-12.50	N	0	>=120 AND >=15 increase from baseline	<=50 AND >=15 decrease from baseline
2	0.00	Y	1	>=120 AND >=15 increase from baseline	<=50 AND >=15 decrease from baseline
3	-2.50	N	0	>=120 AND >=15 increase from baseline	<=50 AND >=15 decrease from baseline
4	6.25	N	0	>=120 AND >=15 increase from baseline	<=50 AND >=15 decrease from baseline
5	7.50	N	0	>=120 AND >=15 increase from baseline	<=50 AND >=15 decrease from baseline

Obs	CRIT1FL	CRIT1FN	CRIT2FL	CRIT2FN
1	N	0	N	0
2	N	0	N	0
3	N	0	N	0
4	N	0	N	0
5	N	0	N	0

ADVS

Obs	STUDYID	USUBJID	VSSEQ	VISITNUM	VISIT	TRT01SDT	TRT01EDT	AVISITN	AVISIT
6	ABC2201	ABC2201_1001001	26	105	WEEK8	2012-08-28	2012-12-11	105	WEEK8
7	ABC2201	ABC2201_1001001	27	106	WEEK12	2012-08-28	2012-12-11	106	WEEK12

Obs	ADT	ADY	ANL01FL	ANL01FN	ANL01DSC	AVAL
6	2012-10-29	63	Y	1	Analysis record flag for Double-Blind Period	77
7	2012-11-23	88	Y	1	Analysis record flag for Double-Blind Period	97

Obs	PARAMCD	PARAM	PARAMN	BASE	CHG
6	STDPULSE	Standing Pulse Rate (BEATS/MIN)	1	80	-3.00
7	STDPULSE	Standing Pulse Rate (BEATS/MIN)	1	80	17.00

Obs	PCHG	ABLFL	ABLFN	CRIT1	CRIT2
6	-3.75	N	0	>=120 AND >=15 increase from baseline	<=50 AND >=15 decrease from baseline
7	21.25	N	0	>=120 AND >=15 increase from baseline	<=50 AND >=15 decrease from baseline

Obs	CRIT1FL	CRIT1FN	CRIT2FL	CRIT2FN
6	N	0	N	0
7	N	0	N	0

Output 2. Derived Analysis Data Set – ADVS

```
%ALTVAL(SILIB=WORK,SDLIB=WORK,SIDB=VS,SDDB=ADVS,IDVARS=%STR(USUBJID VISITNUM),
FRQVAR=%STR(AVISIT*ADT*TRT01SDT*ABLFL*ABLFN  ADT*VSDTC));
```

The above macro call will conduct common variable check and three additional checks. The first check is the cross-frequency check for AVISIT, ADT, TRT01SDT, ABLFL, and ABLFN. If there are any cases with ABLFL="Y" and ADT> TRT01SDT, it suggests a problem since ABLFL is defined as the last non-missing value record prior to or equal to TRT01SDT. The second check is the cross-frequency check for ADT and VSDTC. The format for ADT and VSDTC is different, so it is easy to check the value by just eyeballing them.

Step-by-step SAS outputs are shown below.

The ALTVAL macro first generated the list of common variables in both raw (VS) and derived analysis data set (ADVS), from the common variable check.

COMMON VARIABLES IN BOTH VS AND ADVS

Obs	NAME
1	STUDYID
2	USUBJID
3	VISIT
4	VISITNUM
5	VSSEQ

Output 3. Common Variables in Both VS and ADVS.

Next, the ALTVAL macro generated the list of variables that only exist in the derived analysis data set (ADVS).

VARIABLES IN ONLYDRV

Obs	NAME
1	ABLFL
2	ABLFN
3	ADT
4	ADY
5	ANL01DSC
6	ANL01FL
7	ANL01FN
8	AVAL
9	AVISIT
10	AVISITN
11	BASE
12	CHG
13	CRIT1
14	CRIT1FL
15	CRIT1FN
16	CRIT2
17	CRIT2FL
18	CRIT2FN
19	PARAM
20	PARAMCD
21	PARAMN
22	PCHG
23	TRT01EDT
24	TRT01SDT

Output 4. Variables Only in Derived Analysis Data Set (ADVS).

The ALTVAL macro further generated the list of variables that only exist in the raw data set (VS).

VARIABLES IN ONLYRAW

Obs	NAME
1	DOMAIN
2	VSCAT
3	VSDTC
4	VSORRES
5	VSORRESU
6	VSPOS
7	VSSTRESN
8	VSSTRESU
9	VSTEST
10	VSTESTCD

Output 5. Variables Only in Raw Data Set (VS).

The ALTVAL macro further generated the results of proc compare results for common variables.

Compare common variables between data=work.advs and comp=work.vs

The COMPARE Procedure
Comparison of WORK.SADVS with WORK.FVS
(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs
WORK.SADVS	13JAN14:19:19:30	13JAN14:19:19:30	29	7
WORK.FVS	13JAN14:19:19:30	13JAN14:19:19:30	15	7

Variables Summary

Number of Variables in Common: 5.
Number of Variables in WORK.SADVS but not in WORK.FVS: 24.
Number of Variables in WORK.FVS but not in WORK.SADVS: 10.
Number of ID Variables: 2.
Number of VAR Statement Variables: 5.

Listing of Variables in WORK.SADVS but not in WORK.FVS

Variable	Type	Length	Format	Label
TRT01SDT	Num	8	IS8601DA.	Date of First Exposure to Treatment
TRT01EDT	Num	8	IS8601DA.	Date of Last Exposure to Treatment
AVISITN	Num	8		Analysis Visit (N)
AVISIT	Char	20	$20.	Analysis Visit
ADT	Num	8	IS8601DA.	Analysis Date
ADY	Num	8		Analysis Relative Day
ANL01FL	Char	1	$1.	Analysis Record Flag 01
ANL01FN	Num	8		Analysis Record Flag 01 (N)
ANL01DSC	Char	45	$45.	Analysis Record Flag 01 Description
AVAL	Num	8	BEST12.	Analysis Value
PARAMCD	Char	20	$20.	Parameter Code
PARAM	Char	40	$40.	Parameter
PARAMN	Num	8		Parameter(N)
BASE	Num	8	BEST12.	Baseline Value
CHG	Num	8	8.2	Change From Baseline

Compare common variables between data=work.advs and comp=work.vs

The COMPARE Procedure
Comparison of WORK.SADVS with WORK.FVS
(Method=EXACT)

Observation Summary

Observation	Base	Compare	ID
First Obs	1	1	USUBJID=ABC2201_1001001 VISITNUM=1
Last Obs	7	7	USUBJID=ABC2201_1001001 VISITNUM=106

Number of Observations in Common: 7.
Total Number of Observations Read from WORK.SADVS: 7.
Total Number of Observations Read from WORK.FVS: 7.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 7.

NOTE: No unequal values were found. All values compared are exactly equal.

Output 6. Results of Proc Compare for Common Variables.

ALTVAL macro shows cross frequency check results for AVISIT, ADT, TRT01SDT, ABLFL, and ABLFN.

Cross check variables
for avisit*adt*trt01sdt*ablfl*ablfn

The FREQ Procedure

AVISIT	ADT	TRT01SDT	ABLFL	ABLFN	Frequency	Cumulative Frequency
BASELINE	2012-08-28	2012-08-28	Y	1	1	1
SCREENING	2012-08-21	2012-08-28	N	0	1	2
WEEK1	2012-09-04	2012-08-28	N	0	1	3
WEEK12	2012-11-23	2012-08-28	N	0	1	4
WEEK2	2012-09-11	2012-08-28	N	0	1	5
WEEK4	2012-09-28	2012-08-28	N	0	1	6
WEEK8	2012-10-29	2012-08-28	N	0	1	7

Output 7. Results of Cross Frequency Check.

ALTVAL macro shows cross frequency check results for ADT and VSDTC.

Cross check variables
for adt*vsdtc

The FREQ Procedure

ADT	VSDTC	Frequency	Cumulative Frequency
2012-08-21	2012-08-21	1	1
2012-08-28	2012-08-28	1	2
2012-09-04	2012-09-04	1	3
2012-09-11	2012-09-11	1	4
2012-09-28	2012-09-28	1	5
2012-10-29	2012-10-29	1	6
2012-11-23	2012-11-23	1	7

Output 8. Results of Cross Frequency Check.

Finally, the user can use the ALTVAL macro produced DIFF (or &OUTDB) data set to conduct “the logic check of variable specification”. This approach can check cases with reverse conditions in the variable definition rule. The following is to illustrate how to check the reverse condition: “if ADT=TRT01SDT and ABLFN=1”.

```
DATA CHECK1 (KEEP=MSG USUBJID ADT TRT01SDT ABLFN);
 SET DIFF END=LAST;
ATTRIB MSG FORMAT=$80. LABEL="Validation result";
IF ADT=TRT01SDT AND ABLFN=0 THEN DO;
  MSG="ADT=TRT01SDT AND ABLFN=0";
  MSGNO=1;
 OUTPUT CHECK1;
END;
ELSE IF ABLFN=1 AND ADT>TRT01SDT>. THEN DO;
  MSG="ABLFN=1 AND ADT>TRT01SDT>.";
  MSGNO=1;
 OUTPUT CHECK1;

END;
ELSE DO;
  MSG="No issue found";
```

```
  MSGNO=0;
End;
*note: you can conduct as many similar checks as you want;
If LAST AND MSGNO=0 THEN OUTPUT CHECK1;
RUN;

PROC PRINT DATA=CHECK1;
 TITEL "Check1";
RUN;
```

The user can print out CHECK1 for detail further reference.

Cross check will list all the combination of values for the included variables. It will help further identify potential issues that were not considered in the data derivation specification. It has to be checked each time the program is rerun. If the data set is too big and if too many cross check variables are included, one may encounter an error message indicating out of memory.

The logic check of variable specification using algorithm is reversed from the data derivation specification. It typically yields more concise output. The user may need to write more than one such algorithm in order to fully check the derived variable. But, its advantage is that the results are there and easy to check each time the program is rerun.

PRO AND CONS OF THIS MACRO

The ALTVAL macro proposed in this paper is very easy to implement and can be conducted by an entry level programmer. Most importantly, this macro can be used to replace double-programming for the most critical level of validation and thus can potentially save a lot of time and manpower resources. It can also be used as an alternative to conduct other levels of validation for derived analysis data sets. Specifically, the check is direct and visualized, whether by checking the PROC COMPARE part of common variables, by checking the cross frequency outputs of derived variables, or by the logic check of variable specification. It is good to validate the variables coming from unique raw data set without complicated derivations involved and it can point out whether an error is from the derived analysis data for the common variable check and non-common variable check. For non-common variables with the logic check of variable specification, it will not only generate the output for those observations that could not meet the variable derivation rules based on data derivation specification. It will also show the details by listing the records and/ or variables that were not derived correctly based on the derivation specification. The double programming only provides the discrepancies between a production data set and a validation data set. It will not tell a reviewer which data set is corrected and thus the reviewer has to find the reasons by checking the raw data set and derivation rules. For non-common variables with cross-frequency validation, when reviewer manually reviews the output, it helps provide a full view of the data set which may help the reviewer identify derivation rules that may need to be updated, which cannot be achieved by double programming.

The proposed macro also has several limitations. First, ID variable(s) need to be created if they do not exist in the data sets. Second, when the derived variables in the derived analyses data set are derived from multiple input data sets or some observations are from different data sets, the macro has to be called multiple times to validate the variables. Third, it may be difficult to use this macro if there is any structural change between the raw data set(s) and the derived analysis data set.

CONCLUSION

In summary, the proposed ALTVAL macro is easy to use and provides an alternative approach to conduct validation of derived analysis data set without double programming for the most critical level of validation as well as for the critical and non-critical level of validation. It involves several steps as illustrated in this paper and has several advantages over the traditional double programming approach. Despite this, the proposed alternative approach needs to be tested prior to its wide use.

ACKNOWLEGMENTS

Many thanks to James Gallagher for his supports on the paper. Many thanks to Jacques Lanoue, Gary Moore, and Peter Eberhardt for reviewing and helps on the paper.

CONTACT INFORMATION

Linfeng Xu, MSPH,
Integrated Information Sciences (IIS)
Novartis Pharmaceuticals Corporation
One Health Plaza
East Hanover, NJ 07936-1080
USA
Phone: 973-295-2747(cell)
Davidlfxu2000@yahoo.com

Christina E. Scienski
Associate Director, Statistical Programming
IIS, Neurodegeneration
Novartis Pharmaceuticals Corporation
One Health Plaza, Bldg 135-366
East Hanover, NJ 07936-1080
USA
Phone +1 862 7783126
christina.scienski@novartis.com
www.novartis.com