



INTEGO
GROUP

Paper SI05

STANDARDIZATION = AUTOMATION: A JOURNEY FROM DATASET SPECIFICATION TO SAS CODE

Anastasiia Khmelnytska



Orlando 2020



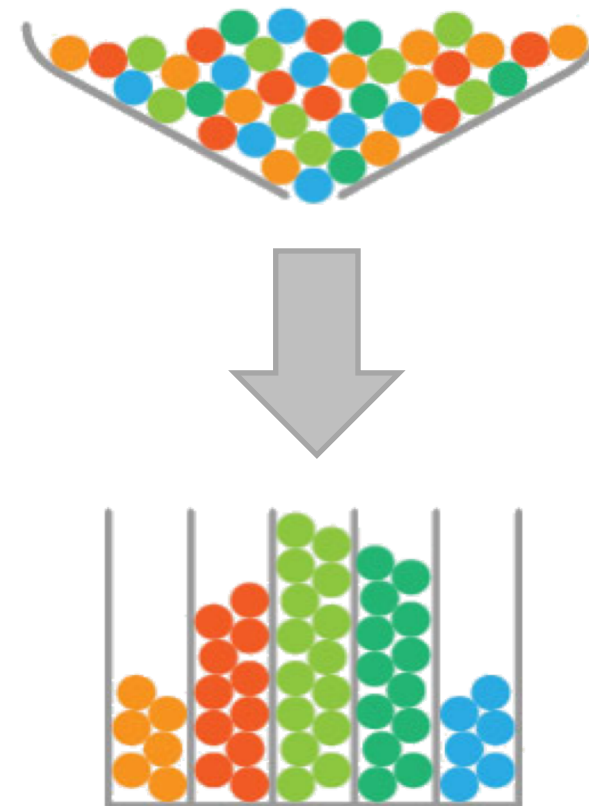
AGENDA

- Overview
- %ATTRIB macro
- %FORMATS macro
- %VLM macro
- Summary



THE NEED FOR AUTOMATION

- Minimizing time for repetitive tasks
- Shorter timelines, faster filings
- Standards implementation made easy
- Interoperability between studies
- Less chances for human errors
- Less code updates





PART I

%ATTRIB MACRO



%ATTRIB MACRO: OVERVIEW

- Perform validity and conformance checks
- Assign variables' attributes based on specification
- Read in dataset metadata

```
%macro attrib (type      = , /*Data type: SDTM or ADAM*/  
                spec      = , /*Location of specifications file*/  
                domain    = , /*Name of SDTM or ADAM dataset that  
                               you need metadata for*/  
                dsnin     = , /*Name of input dataset*/  
                outlib    =  /*Libname for the location where  
                               final dataset will be stored*/);
```

CONFORMANCE CHECKS

- Check that DOMAIN value is valid according to CDISC standards

```
%if &type. = SDTM %then %do;  
  %if not (&domain. in AE ... VS) %then %do;  
    %put WARNING: &domain. is not a valid SDTM domain name.;  
    %return;  
  %end;  
%end;  
%else %if &type. = ADAM and (%substr(&domain., 1, 2) ^= AD  
%then %do;  
  %put WARNING: &domain. is not a valid ADAM dataset name.;  
  %return;  
%end;
```

VARIABLES' ATTRIBUTES: IMPORT

- Name
- Label
- Type
- Length

1	Variable Name	Variable Label	Type	Core	Length
2	STUDYID	Study Identifier	Char	Req	13
3	DOMAIN	Domain Abbreviation	Char	Req	2
4	USUBJID	Unique Subject Identifier	Char	Req	24
5	VSSEQ	Sequence Number	Num	Req	8
6	VSTESTCD	Vital Signs Test Short Name	Char	Req	8
7	VSTEST	Vital Signs Test Name	Char	Req	40
8	VSORRES	Result or Finding in Original Units	Char	Exp	8

```
proc import datafile = &spec. dbms = xlsx replace
      out          = &domain._attrib (keep = variable_name
                                     variable_label type length);

      sheet = "&domain.";

run;
```



VARIABLES' ATTRIBUTES: ASSIGNMENT

- List of variables to keep is stored in VAR macro variable
- Attrib statement is produced and stored in ATTRIB macro variable

```
data _null_;
  set &domain._attrib;
  call symput ("VAR", symget("VAR") || " " || strip(variable name));
  if type = "Char" then stype = "$";
  call symput ("ATTRIB", symget("ATTRIB") || " "
    || strip(variable_name) || " label='" || strip(variable_label)
    || "' length=" || stype || strip(put(length, 3.)));
run;
```

- CALL SYMPUT and SYMGET should be used instead of %let and &

VARIABLES’ ATTRIBUTES: RESULT

Variable Name	Variable Value			
VAR	STUDYID DOMAIN USUBJID VSSEQ VSTESTCD VSTEST VSORRES			
ATTRIB	attrib	STUDYID	label='Study Identifier'	length=\$13
		DOMAIN	label='Domain Abbreviation'	length=\$2
		USUBJID	label='Unique Subject Identifier'	length=\$24
		VSSEQ	label='Sequence Number'	length= 8
		VSTESTCD	label='Vital Signs Test Short Name'	length=\$8
		VSTEST	label='Vital Signs Test Name'	length=\$40
		VSORRES	label='Result or Finding in Original Units'	length=\$8

DATASET METADATA

1	Dataset	Description	Class	Structure	Keys
18	VS	Vital Signs	Findings	One record per vital sign measurement per time point per visit per subject	STUDYID, USUBJID, VSTESTCD, VISITNUM, VSTPTNUM

- Keep dataset label and sort order in LABEL and KEYS macro variables

```
proc import datafile = &spec. dbms = xlsx replace
    out = &domain._sort (keep = dataset description keys);
    sheet = "Dataset Metadata";
run;

data _null_;
    set &domain._sort (where = (dataset = "&domain."));
    call symput("keys", compress(keys, ","));
    call symput("label", strip(description));

run;
```



BRINGING IT ALL TOGETHER

- Output final dataset, assign variables' attributes, create dataset label and keep the necessary variables, remove formats if needed

```
data &outlib..&domain. (label = "&label." keep = &var.);  
  &attrib.;  
  set &dsnin.;  
  %if &type. = SDTM %then %do;  
    format _all_;  
  %end;  
run;
```

- Sort the dataset by its unique key

```
proc sort data = &outlib..&domain.;  
  by &keys.;  
run;
```



PART II

%FORMATS MACRO



%FORMATS MACRO: OVERVIEW

- Derive variables that only need formatting
- Examples: --TESTCD, --TEST, --METHOD, VISIT, VISITNUM, PARAM
- Either CDISC codelists or your own
- Just add “Controlled Terminology” spreadsheet to your specification

```
%macro formats(domain = , /*Name of the domain for which you  
                                want to create formats*/  
                spec    = /*Location of specifications file*/);
```

CONTROLLED TERMINOLOGY EXAMPLE

Domain	Codelist	Reported value	Submission value	Domain	Codelist	Reported value	Submission value
MULTIPLE	VISITNUM	SCRNBASE	-30	VS	VSTPT	0	PREDOSE
MULTIPLE	VISITNUM	DAY1	1	VS	VSTPT	1	1 HOUR POSTDOSE
MULTIPLE	VISITNUM	DAY2	2	VS	VSTPT	2	2 HOURS POSTDOSE
MULTIPLE	VISITNUM	DAY3	3	VS	VSTPT	3	3 HOURS POSTDOSE
MULTIPLE	VISITNUM	DAY4	4	VS	VSTPT	4	4 HOURS POSTDOSE
MULTIPLE	VISITNUM	DAY5	5	VS	VSTPT	6	6 HOURS POSTDOSE
Domain	Codelist	Reported value	Submission value	Domain	Codelist	Reported value	Submission value
VS	VSTESTCD	VSBPDIA	DIABP	VS	VSTEST	DIABP	Diastolic Blood Pressure
VS	VSTESTCD	HEIGHT	HEIGHT	VS	VSTEST	HEIGHT	Height
VS	VSTESTCD	VSPULSE	PULSE	VS	VSTEST	PULSE	Pulse Rate
VS	VSTESTCD	VSRESP	RESP	VS	VSTEST	RESP	Respiratory Rate
VS	VSTESTCD	VSBPSYS	SYSBP	VS	VSTEST	SYSBP	Systolic Blood Pressure
VS	VSTESTCD	VSTEMP	TEMP	VS	VSTEST	TEMP	Temperature
VS	VSTESTCD	WEIGHT	WEIGHT	VS	VSTEST	WEIGHT	Weight

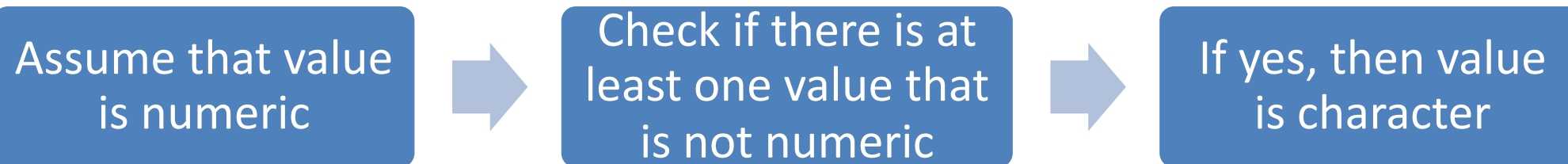
CREATING FORMAT FROM A DATASET

- Dataset must include three mandatory variables:
 - FMTNAME – name of the format
 - START – variable that contains the “from” value
 - LABEL – variable that contains the “to” value
- Another important variable: TYPE

Value	Stands for	Compatible with	Converts from	Converts to
C	Character format	PUT function	Character	Character
N	Numeric format	PUT function	Numeric	Character
J	Character informat	INPUT function	Character	Character
I	Numeric informat	INPUT function	Character	Numeric
P	Picture format	PUT function	Numeric	Character

DETERMINING THE TYPE

- First determine types of reported value and submission value



- Then, based on the previous table assign the value of TYPE

```

data fmt2;
  merge fmt fmt_type;
  by domain codelist;
    if type_rep = "char" and type_ct = "char" then type = "c";
  else if type_rep = "char" and type_ct = "num" then type = "i";
  else if type_rep = "num" and type_ct = "char" then type = "n";
run;
  
```

FORMATS OUTPUT AND USE

- Output formats using CNTLIN option of PROC FORMAT

```
proc format cntlin = fmt2 (rename = (codelist = fmtname
                                     repvalue = start
                                     ctvalue   = label));
```

- Create the needed variables in your program with a few lines of code

```
data vs;
  set raw.vs;
  vstestcd = put(vsnam, $vstestcd.);
  vstest    = put(vsnam, $vstest.);
  visitnum = input(visid, visitnum.);
  vstpt     = put(tpt, vstpt.);

run;
```



PART III

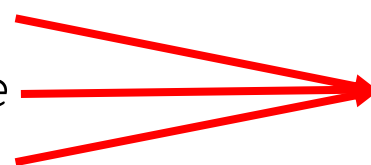
%VLM MACRO



%VLM MACRO: OVERVIEW

- Value-level metadata can be used when variables' derivations depend on the values of other variables
- Example: value of AVAL may be derived differently for each PARAMCD
- If-then conditional statements are created based on value-level metadata and stored in VLM macro variable

```
data _null_;
  set vlm (where = (dataset="&domain.")) ;
  first-block-of-code
  second-block-of-code
  third-block-of-code
run;
```



if **where-condition** then **variable** =
algorithm-dependent-derivation

EXAMPLE 1: SIMPLE ASSIGNMENT (1)

DATASET ▾	VARIABLE ▾	WHERE ▾	ALGORITHM ▾	CTVALUE ▾	DATA TYPE ▾	LENGTH ▾
ADLB	PARAM	LBCAT='CHEMISTRY' and LBTESTCD='ALB'	Set to value in CTVALUE	Albumin (g/dL)	text	40
ADLB	PARAM	LBCAT='CHEMISTRY' and LBTESTCD='ALP'	Set to value in CTVALUE	Alkaline Phosphatase (U/L)	text	40
ADLB	PARAM	LBCAT='CHEMISTRY' and LBTESTCD='ALT'	Set to value in CTVALUE	Alanine Aminotransferase, ALT	text	40
ADLB	PARAM	LBCAT='CHEMISTRY' and LBTESTCD='AST'	Set to value in CTVALUE	Aspartate Aminotransferase, AST	text	40
ADLB	PARAM	LBCAT='HEMATOLOGY' and LBTESTCD='BASO'	Set to value in CTVALUE	Basophils (10 ⁹ /L)	text	40
ADLB	PARAM	LBCAT='CHEMISTRY' and LBTESTCD='BIL'	Set to value in CTVALUE	Total Bilirubin (mg/dL)	text	40

- PARAM has different values based on the corresponding values of LBCAT and LBTESTCD
- It is assigned a value from another column CTVALUE
- An example of if-then statement generated:

```
if LBCAT='CHEMISTRY' and LBTESTCD='ALB'
  then PARAM=put("Albumin (g/dL)",40.);
```

EXAMPLE 1: SIMPLE ASSIGNMENT (2)

- First block of code in the DATA _NULL_ step of VLM macro based on the value of ALGORITHM column

```
if algorithm = "Set to value in CTVALUE" then do;
  call symput ("vlm", symget("vlm") || ' if ' || strip(where)
              || " then " || strip(variable) || "=");
  if datatype="text" then call symput ("vlm", symget("vlm")
              || 'put("' || strip(ctvalue) || "', '
              || strip(put(length, 3.)) || "."); ");
  else call symput ("vlm", symget("vlm") || strip(ctvalue) || "; ");
end;
```

- Either put statement or simple assignment are used depending on variable type

EXAMPLE 2: ROUNDED VALUE (1)

DATA SET	VARIABLE	WHERE	ALGORITHM	DATA TYPE	LENGTH	NUMB_DEC_PLACES
ADLB	AVAL	LBCAT='CHEMISTRY' and LBTESTCD='ALB'	Set to value of LB.LBSTRESN with specified number of decimals	float	8	1
ADLB	AVAL	LBCAT='CHEMISTRY' and LBTESTCD='ALP'	Set to value of LB.LBSTRESN with specified number of decimals	float	8	0
ADLB	AVAL	LBCAT='CHEMISTRY' and LBTESTCD='ALT'	Set to value of LB.LBSTRESN with specified number of decimals	float	8	0
ADLB	AVAL	LBCAT='CHEMISTRY' and LBTESTCD='AST'	Set to value of LB.LBSTRESN with specified number of decimals	float	8	0
ADLB	AVAL	LBCAT='HEMATOLOGY' and LBTESTCD='BASO'	Set to value of LB.LBSTRESN with specified number of decimals	float	8	2
ADLB	AVAL	LBCAT='CHEMISTRY' and LBTESTCD='BILI'	Set to value of LB.LBSTRESN with specified number of decimals	float	8	1

- AVAL is derived as LBSTRESN rounded to a specified number of decimals
- An example of if-then statement generated:

```
if LBCAT='CHEMISTRY' and LBTESTCD='ALB'
then AVAL=round(LBSTRESN,0.1);
```

EXAMPLE 2: ROUNDED VALUE (2)

- Records with the specified pattern of the ALGORITHM column are identified using PRXMATCH function

```
else if prxmatch("/(Set to value of \S+ with specified number of
decimals)/", algorithm) then do;
  round = 0.1 ** numb_dec_places;
  call symput ("vlm", symget("vlm") || ' if ' || strip(where) || ' then '
    || strip(variable) || '=round(' || strip(scan(substr(algorithm, 17),
      2, ". ")) || ', ' || strip(put(round, best.)) || '); ');
end;
```

- Temporary variable ROUND is created to transform number of decimal places to the expected second argument of ROUND function

EXAMPLE 3: MULTIPLICATION (1)

DATA SET ▾	VARIABLE ▾	WHERE ▾	ALGORITHM ▾	DATA TYPE ▾	LENGTH ▾
LB	LBSTRESN	TESTNAME='GLUCOSE' and UNITS = 'MG/DL'	Set to value of LBORRES * 0.0555	num	8
LB	LBSTRESN	TESTNAME='HDL-CHOLESTEROL'	Set to value of LBORRES * 0.0259	num	8
LB	LBSTRESN	TESTNAME='HEMATOCRIT'	Set to value of LBORRES * 0.01	num	8
LB	LBSTRESN	TESTNAME='HEMOGLOBIN'	Set to value of LBORRES * 10	num	8
LB	LBSTRESN	TESTNAME='IRON'	Set to value of LBORRES * 0.1791	num	8
LB	LBSTRESN	TESTNAME='LDL-CHOLESTEROL'	Set to value of LBORRES * 0.0259	num	8

- LBSTRESN is set to the value of LBORRES multiplied by some conversion factor
- An example of if-then statement generated:

```
if TESTNAME='GLUCOSE' and UNITS = 'MG/DL'  
  then LBSTRESN=input(LBORRES,best.) * 0.0555;
```

EXAMPLE 3: MULTIPLICATION (2)

- Using PRXMATCH we select the following pattern in ALGORITHM column:
“Set to value of *some-variable-name* * *conversion-factor*”

```
if prxmatch("/(Set to value of \S+ \* \d+)/", algorithm) then do;
  if prxmatch("/(\w+\.\w+ \*)/", algorithm)
    then var_orig = scan(substr(algorithm, 17), 2, ". ");
  else var_orig = scan(substr(algorithm, 17), 1, " ");
  call symput ("vlm", symget("vlm") || ' if ' || strip(where) || ' then '
    || strip(variable) || '=input(' || strip(var_orig) || ',best.) * '
    || scan(algorithm, -1, " ") || '; ');
end;
```

- With the second PRXMATCH we determine whether there was a two-level variable name in the specification (DOMAIN.VARIABLE)

USE OF VLM MACRO VARIABLE

- Call %VLM macro in your dataset program and invoke VLM macro variable in the appropriate data step
- It will resolve to all of the statements that were created from value-level metadata and you will have your code generated for you

```
data lb;
  set rawdata.labs;
  some-statements
  &vlm.;
  some-more-statements

run;
```

Note if-then statements order

P R E S E N T A T I O N T A K E A W A Y S

- Possibilities for automation are everywhere
- Need for automation is growing
- Most common areas for automation
- Macros can be used directly or customized
- Detailed description of development process allows you to create similar macros of your own



THANK YOU

Anastasiia Khmelnytska
Intego Group LLC
19 Hromadyanska Street
Kharkiv 61057, Ukraine
Work Phone: +1 407.512.1006 (ext. 2443)
Email: anastasiia.khmelnytska@intego-group.com
Web: www.intego-group.com

