PhUSE US Connect 2018

Paper CT17

CODING TIPS AND TRICKS FOR DEFINE VLM CODELISTS

Charlie Sowers, IQVIA, Durham, USA

ABSTRACT

Define XML submissions require metadata descriptions of derived variables, organized by parameter. Variables are divided into three categories, Integer, Float, and Character, then can be further subdivided based on length and derivation. Character variables are divided in a certain way: into named codelists containing every potential character option for that variable within a list of parameters. The goal of this paper is to provide insights on potential techniques for creating codelists for character variables and naming these codelists.

INTRODUCTION

For the purposes of this paper we will be using the formats used by the Pinnacle 21 Define generator. The portion of the Define excel document that is used to handle the creation of the Value Level Metadata codelists comes from three tabs: the Valuelevel, Whereclauses, and Codelists tabs. The Valuelevel tab will have a row for each grouping of variable values. The Whereclauses column of the Valuelevel tab directs to the ID column of the Whereclauses tab, which describes the Parameter code values that contain the same metadata within the variable. The Valuelevel tab similarly has a codelist column that directs to the ID column of the Codelists tab, which will have a row for each possible value of the codelist. The Whereclauses tab and the codelist tab are thus linked through the Valuelevel tab. To automate the process of creating these codelists it is thus necessary to create a simple connection between the list of parameters provided in the Whereclauses tab, the variable name in question, and the codelist names.
We suggest the following solution is to first alphabetize within each variable by lowest alphabetical PARMCD (as listed in the whereclauses tab). Once they are ordered we translate the order number into character very simply: the first is assigned AA, the second AB, and so on. The final codelist name will be a combination of the dataset name, variable name, and character assignment. For example, 'ADVS-CRIT1-CL-AA' would be the name of the first codelist for CRIT1 in ADVS ('CL' just stands for Codelist').

EXAMPLE DEFINE EXCEL DOCUMENT CODELIST INFORMATION:

Valuelevel Tab:

Dataset	Variable	Where Clause	Codelist
ADVS	CRIT1	ADVS.CRIT1.PSDIABP-STDIABP	ADVS-CRIT1-CL-AA
ADVS	CRIT1	ADVS.CRIT1.PSPULSE-STPULSE	ADVS-CRIT1-CL-AB
ADVS	CRIT1	ADVS.CRIT1.WEIGHT-WEIGHTS	ADVS-CRIT1-CL-AC

Whereclauses tab:

ID	Dataset	Variable	Comparator	Value
ADVS.CRIT1.PSDIABP-STDIABP	ADVS	PARAMCD	IN	PSDIABP, SDDIABP, STDIABP
ADVS.CRIT1.PSPULSE-STPULSE	ADVS	PARAMCD	IN	SDPULSE, SPPULSE, STPULSE, PSPULSE
ADVS.CRIT1.WEIGHT-WEIGHTS	ADVS	PARAMCD	IN	WEIGHT, WEIGHTS

Codelists tab:

ID	NAME	Data Type	TERM
ADVS-CRIT1-CL-AA	ADVS CRIT1 Codelist AA	text	>=105 mmHg and >=15 mmHg increase
ADVS-CRIT1-CL-AB	ADVS CRIT1 Codelist AB	text	>=120 bpm and >=15 bpm increase
ADVS-CRIT1-CL-AC	ADVS CRIT1 Codelist AC	text	>=7% increase

CODELIST CREATION:

Automating the creation of the value level codelists overall is not a conceptually challenging task, and the implementation does not require any tricks. In overall terms, it is easy to find for each dataset-variable-parameter combination the list of potential unique character values. Once you have done this, create list of parameters with the same possibilities. In SAS®, this can be done with a simple transpose, creating a column for each potential value of every parameter-variable combination in alphabetical order. Sorting by the dataset and variable name, then the 1st such column, then the second, and so on, will group the parameters with those that have the exact same values. From here a Retained variable can create a list of parameters within a single value. In SAS® code:

```
*once you have only 1 row per possible character value within each Parameter and Variable;

data codelist;
  set codelist;
  by dataset var_name paramcd val;
  retain valnum;
  if first.paramcd then valnum = 1;
  else valnum = valnum+1;
run;

proc transpose data = codelists out = codeltrn prefix = val;
  by dataset var_name paramcd;
  var val;
  id valnum;
run;

proc sort;
  by dataset var_name val:; *using a colon after val ensures that you sort by every column val1, val2, etc.;
run;
```

From here it is simple to make only one row for each codelist with the parameter codes in an alphabetical order in a single column.

NAMING THE CODELISTS:

Assigning the alphabetic names based on the list of parameter codes requires a simple trick. After the above step, it is useful to create a character variable that is just the entire alphabet in a string. Then division+the floor function can be used for the first letter of the alphabetic assignment, and the Mod function for the second letter. The Mod function essentially gives the remainder after integer division, and the Floor function along with division is essentially recreating integer division. In SAS®:

```
data codelist;
  set codelist;
  retain ord;
  by dset varnam paramlst;

  alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

  if first.paramlst then ord = 0;
  else ord = ord+1;

  letters = substr(alphabet,floor(ord/26)+1, 1) || substr(alphabet, mod(ord, 26)+1, 1);

run;
```

This will store the ‘AA’, ‘AB’ etc. assignments in the variable LETTERS. This can be used to create the final codelist names that are simple, readable, and unique to each codelist as seen above.

CONCLUSION

There is no one correct way to automate the process of creating and naming Define codelists for character variables. The method and naming convention outlined above is a straightforward method of organizing the automation of this

process, with examples of how to solve the more difficult problems outlined in SAS® code in hopes that they will help the reader to implement these steps.

CONTACT INFORMATION

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

Author Name: Charlie Sowers
Company: IQVIA
City / Postcode: Durham, USA
Email: Charlie.sowers@iqvia.com