PhUSE 2016

Paper CS06

Coding WHO-DDE: an automatic, easy and powerful SAS tool

Gianluca Mortari, CROS NT, Verona, Italy

ABSTRACT

Concomitant medication coding process is a great challenge for data managers having to deal with it in multicentric clinical trials. It is a time-consuming process and finding an appropriate approach to WHO-DDE™ dictionaries can be difficult. This paper is intended to show a powerful SAS® tool for Medication coding that can handle all the phases of the process, from the cleaning of the reported name of the treatment (Verbatim Term) to the coding, with an interactive GUI approach showing best coding choices. This tool also includes some useful features like stemming algorithms between the dictionary term and the term reported as well as for handling typos and other kind of inconsistencies. The possibility to save and exit from the work process anytime is included along with automatic data driven dictionary filtering and much more. This tool can be used by anyone, even those without any SAS knowledge, and it is intended to optimize the process flow.

INTRODUCTION

The main goals that drove the creation of a SAS tool for coding Concomitant Medications with the WHO-DDE™ dictionary can be classified as follows:

1) Easy usage and limited interaction with SAS code: data managers are not usually experienced SAS programmers, so they do not need to understand the content of the program. For this reason I created the program so that the only interaction with the SAS code would be setting the parameters for the macro.
2) Powerful in cleaning and coding: data managers spend a great amount of their working time with mechanical tasks, and it is better to let SAS do it for them. Data managers will be able to forget about copy-pasting from dictionaries.
3) Fast and innovative: speeding up the coding process, saving time for other tasks. Innovative in its programming perspective, using the output window as a GUI and managing the dictionary dataset with advanced SAS macro code.

The purpose of this paper is to give an overview of the experience of the creation of this automatic and interactive tool strictly related to SAS programming. The handling of the WHO-DDE dictionaries will not be covered here, as it is well documented online and a relatively common topic for papers. This means that even if this tool is focused on the structure of the WHO-DDE dictionaries, the same approach with windows and user interactions can be easily applied to other fields, related to dictionaries or not.

WHAT YOU NEED:

The environment setup requires the following components:

1) The Global Dictionary Datasets: these will be derived by the programmer directly from the original Uppsala Monitoring Centre website, at the World Health Organisation Drug Dictionary Enhanced page. CHEM and MED datasets, used in this paper need to have a single observation for each Drug Name associated with each Preferred Name in the first one and each Preferred Name associated with ATC coding in the second one. Basically, all the possible codes for each Drug Name will be present in the huge dataset resulting from the Cartesian product between these datasets. The obvious great dimension of these datasets will necessitate some efficient choices of programming to reduce the computational time between each iteration of the algorithm. For example, we will prefer hashes rather than simple DATA step merges and advanced macro functions rather than feeding the PDV.
2) Raw Dataset of medications to be coded: the main interest will go on the variable containing the Reported Term by the investigator, but we will use some more variables to define the bundle of key identifiers. These variables usually are the Indication and the Route, but in some cases we used the Form too. These variables will identify the uniqueness of each observation, this means that if the same sequence of key variables will be repeated more than once in the same dataset, they will be handled as a single term to be coded even if the final structure of the coded dataset will be the same as the starting one. The raw dataset that will be used as an example in this paper contains 200 observations related to a study on pregnancy control.
3) The Working Library: this is the folder where all the dataset useful for choosing the correct coding will be created, the Data Manager can look at this folder anytime to consult dictionaries and see how the coding process is going, even if we usually call this folder WORK it shouldn't be confused with the SAS work

temporary library. The temporary library will be used to store some datasets necessary for the process but will be deleted at each step.

4) The SAS system. The version we used is 9.2M0 but it is compatible with SAS 9.4 changing the output window from HTML to listing.

PREPARATION AND FINALIZATION, THE STRUCTURE OF THE TOOL

PARAMETERS OF THE MACRO

There are few parameters required to set up the macro call in order to start the procedure:

```
%atc_code(     RAWDS   = data.cm
             , DRUGVAR = drugname
             , ROUTE   = route
             , NDIC    = ind
             , FORM    = NULL
             , WORKLIB = coding
             );
```

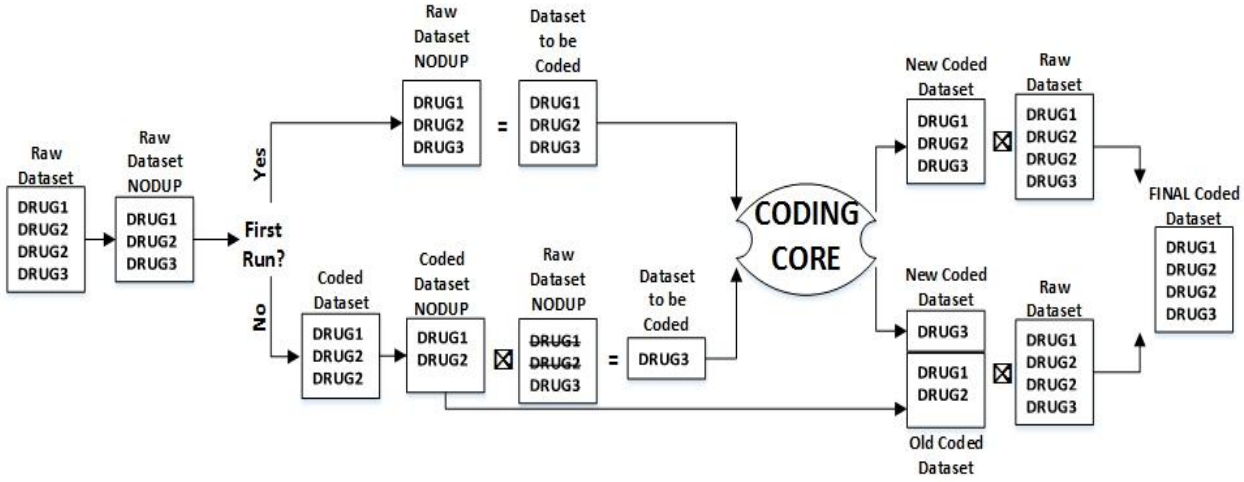
They are quite self-explanatory:

1) DRUGVAR, ROUTE, INDIC and FORM are the previously discussed key variables, the FORM variable can be set to NULL if not present in the database.
2) The RAWDS parameter containing the raw dataset to be coded must be referenced with the two-level name of the form libref.filename, the libref will also be used at the end of the process to save the resulting coded dataset.
3) The WORKLIB parameter refers to the previously discussed Working Library.

After launching the macro, some basic checks will be done to assess the compatibility of the parameters defined in the call to prevent the Data Manager from running an incorrect procedure that would lead to wrong results. We will check if `&RAWDS` is reported with the two-level notation and if the corresponding dataset and libref exist in the actual SAS environment as well as the libref of `&WORKLIB`. Then a check on the existence and the character nature of the key variables is performed.

THE STRUCTURE OF THE PROGRAM

The diagram below is intended to describe the structure of the program without the core system developed for the coding that will be discussed in the next chapter.



While this diagram may seem confusing, the concepts are explained throughout this paper. It is important at the beginning of this tool to give an overview of what stands behind the program even if it won't be the interactive core of the coding.

IS THIS THE FIRST RUN? PREPARATION OF THE DATASET TO BE CODED

In some cases, Data Managers need to split the coding activity in multiple steps, for example in the case of incomplete data or when switching to other tasks within the day. For this reason I decided to program inside the macro the possibility to save and exit from the process anytime, and I will cover this later. This will also have an impact at the beginning of the program, during the setup phase discussed in this chapter. When reading the raw dataset, the possible presence of previously coded datasets needs to be taken into account. They should be saved

in the same folder and, in this case, merge the raw dataset with the coded one to handle only those terms still uncoded.

At the end of the procedure we will save the coded dataset just adding the string "_CODED" to the name of the raw dataset, so we can check the presence of a previous coding phase with the *exist()* function, setting up the macro variable &FIRSTRUN to Y if it seems to be the first run, N if not.

```
/* Derive if coding program has been run before and there is already a coded dataset present */
%if %sysfunc(exist(&RAWLIB..&outds._CODED)) = 0 %then %do;
        %let FIRSTRUN=Y;
%end;
%else %do;
        %let FIRSTRUN=N;
%end;
```

Based on the value of the macro variable &FIRSTRUN we can decide how to proceed. If this is the first run, the decision will be to go on simply working on a copy of the raw dataset and start the coding from the beginning. Otherwise we can work on a subset of the raw dataset identified only on those Verbatim Terms still uncoded (new records: where the coding variables are blanks, or those observations skipped in previous coding phases: where coding variables contain the string placeholder "£NOT CODED£").

FINALIZATION

As in the preparation of the dataset to be coded, the value of the variable &FIRSTRUN will have a great impact on the process flow for the finalization which is reached if the coding process ends or if the user decides to save and exit. In both these cases, we can then run the final steps of the macro for the creation of the final coded dataset. The main difference related to the value of the &FIRSTRUN macro variable will be the decision whether to append or the old coded terms which were removed in the initial phase from the previously coded dataset or not.

For simplicity, in the core coding function we end up attaching to each coded term only the singular complete ATC code that identifies only one possible structure for the intermediate levels of coding. For this reason, during the finalization of the coded dataset, we will merge the complete coding to the last level ATC code.

```
 /* Merge on all levels of atc */
data &RAWLIB..&outds._CODED;
        length DGATC1C DGATC2C DGATC3C DGATC4C $7 DGATC1D DGATC2D DGATC3D DGATC4D atc_text $50
        level $1;
        if _n_=1 then do;
                declare hash final(dataset:'codeatce.all_atc');
                final.definekey('atc' );
                final.definedata('level', 'atc text', 'DGATC1C', 'DGATC2C', 'DGATC3C', 'DGATC4C',
                'DGATC1D', 'DGATC2D', 'DGATC3D', 'DGATC4D');
                final.definedone();
        end;
        set finalizer2;
        rc=final.find();
run;
```

The previous code contains a hash merge. Hash objects are an efficient look-up tool in the SAS DATA step, object-oriented programming structures that function differently from traditional SAS language statements. They were implemented from SAS 9 and rather than the usual SAS DATA steps they are memory-resident searching methods not primarily based on comparison between keys, but use a completely different direct-addressing methodology able to speed up the computational time. This is the easiest hash-merge present in this tool and, even if it isn't of significant relevance at this point, I wanted to report it here to give the more basic code to reproduce it. Many papers are present on the web about the wide range of usage of this programming methodology.

If the coding phase reached its natural end with the coding of the last observation present in the dataset to be coded, then these steps will be normally run exiting from the phase 2 loop discussed in the next chapter. Otherwise, if the user decided to save and exit from the process flow, writing the command line in the interactive window, the macro code *%GOTO* will be used to skip all the iterations of the subsequent loops.

```
%if %upcase(%sysfunc(compress("&final_atc."))) = "SAVEANDEXIT" %then %do;
        %goto SAVEANDEXIT;
%end;
```

That will exit from all the loops, skipping the rest of the code, and finally recall the last macro statement:

```
%SAVEANDEXIT: %create_final;
```

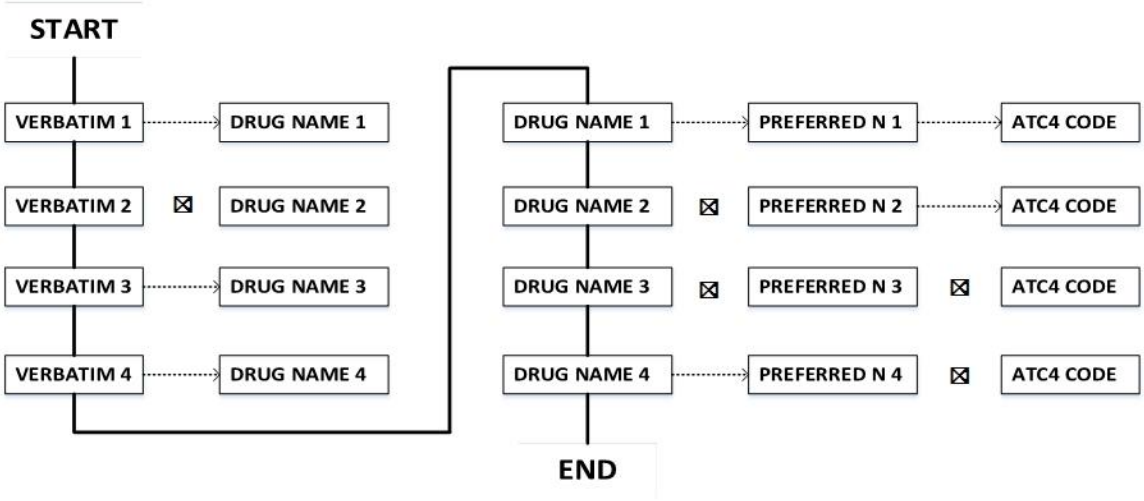
The *%GOTO* function is not widely used in SAS macro programming but it is extremely useful where a complex, robust and flexible tool needs to be programmed. The only limit in its usage is that it should refer to a point inside the same macro where it is executed and it cannot go to a point inside a *%DO* loop of any kind (*%DO*, *%DO %UNTIL*, *%DO %WHILE* and so on). The *%GOTO* macro statement is generally used to terminate a macro leading to an error. The referring label (in this case *%SAVEANDEXIT*) should be followed by a colon.

This final dataset will be saved both in the starting libref with the name of the raw dataset adding the string "_CODED" and in the previously discussed Working Library as a backup, adding the date of creation to the name and a sequential number increasing if within the day more executions of the coding tool were run. Then the name will be "`&WORKLIB..&outds._CODED_&DSETDT._&RUN`" where the macro variable `&DSETDT` is simply the system macro variable `&sysdate` in *yymmdd6.* format and `&RUN` is obtained from a proc contents on the Work Library folder, calculated as the (number of datasets with the name root `&OUTDS`._CODED_`&DSETDT` +1).

THE CODING MACRO

The real core of this paper is the interactive approach to coding that will be discussed in this chapter. The structure of the coding procedure is divided into two different and self-standing phases.

The first one will try to help the Data Manager to associate a Drug Name to each Verbatim Term (the free text reported in the CRF) where it doesn't have a perfect match with the dictionary. The second phase will let the Data Manager decide which is the most correct coding choice displaying the available solutions. The Diagram below describes the core process of coding, the symbol identifies where an interaction required.



A dataset named BUILD will be saved in the Work Library defined as a parameter in the macro call, this will have exactly the same structure of the original dataset but at each action undertaken with the interactive window, the results will be attached allowing the Data Manager to keep trace of the complete coding process and what is happening. The BUILD dataset, as all the other datasets stored in the Work Library, will not be used by the tool in any case, they are stored there only to be consulted by the Data Manager. This is important because if the Data Manager will leave a dataset open while interacting with the output window, the results will not be attached but the action won't be lost and the datasets will be completely refreshed after the next action.

PHASE 1: VERBATIM TERM TO DRUG NAME

This phase of the coding, also named the cleaning phase, tries to match the Verbatim Term with a Drug Name on the dictionary, this is not a simple task as it usually requires the knowledge of each specific drug found in the database to clean the typos or other inconsistencies where present. For this reason, I implemented a stemming algorithm that calculates an index of distance between the reported term and each Drug Name in the database, the lower the value of the index the nearer the dictionary string should be with the Verbatim Term.

First of all, we need to identity in the dataset ready to be coded those texts that don't have a perfect match with the dictionary, remembering that in some cases the same Verbatim Term can be associated with more than one Drug Name, for this purpose the variable HLC was derived in the dictionary. Multiple strategies can be used to obtain the number of observations of this dataset containing only those terms to be matched with the dictionary, this is one:

```
%let dsid=%sysfunc(open(withdrug_spell));
%if &dsid > 0 %then %do;
        %let nobs=%sysfunc(attrn(&dsid,nobs));
        %let rc =%sysfunc(close(&dsid));
%end;
```

LOOPING ON VERBATIM TERMS TO BE CLEANED

A *%DO* loop will start an iteration on each of this observations, inside of which I will obtain all the information related to the specific record to be inserted in the header of the output window. The lengths of the texts will be found to apply the *%substr()* function only when it will exceed the space allocated to that string, otherwise a warning would be produced in the log. An alternative solution to this could be switching to the *subpad()* DATA step function, not directly available in SAS Macro code, but easily applicable trough the *%sysfunc()* exploit. At each iteration all the information regarding the specific Verbatim Term on which the tool is working on will be put into macro variables on a DATA step with multiple call symputx statements. All these variables obtained were predefined as *%GLOBAL* at the beginning of the macro.

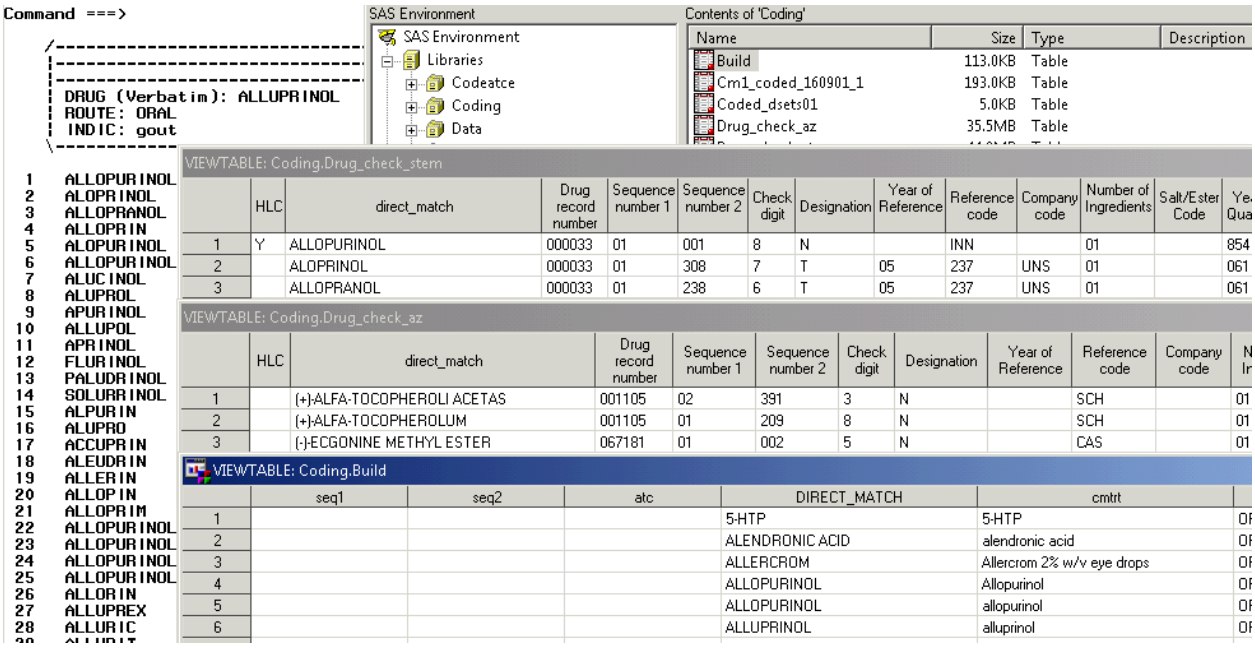
GUESSING THE CORRECT DRUG NAME

Let's proceed with the calculation of stemming indexes and the sorting of the dictionary based on this derived value.

```
/* For each verbatim term to be coded, create variable which show how similar the verbatim term
is to direct match in dictionary */
data chem_t2(drop=repterm_nospc direct_match_nospc);
        set chem_t;
        repterm="&valuez.";
        repterm nospc=compress(repterm);
        direct_match_nospc=compress(direct_match);
        distance1=complev(direct_match_nospc,repterm_nospc);
        distance2=compged(direct match nospc,repterm_nospc);
        drugt=subpad(direct_match,1,45);
run;
proc sort data=chem_t2;
        by distance1 distance2;
run;
```

"The COMPGED function returns the generalized edit distance between two strings. Specifically, the COMPGED function returns a generalization of the Levenshtein edit distance, which is a measure of dissimilarity between two strings. The Levenshtein edit distance is the number of operations (deletions, insertions, or replacement) of a single characters that are required to transform string-1 into string-2."
"The COMPLEV function is very similar to the COMPGED function. The difference is that the Levenshtein edit distance that is computed by COMPLEV is a special case of the generalized edit distance that is computed by COMPGED. The result is the COMPLEV executes much more quickly than COMPGED. However, the COMPLEV function is not as powerful or versatile as the COMPGED function. The COMPLEV function is generally most useful when comparing simple strings and when speed of comparison is important."

After some tries, a good guessing was identified with this combination of indexes, other useful functions to improve the matching power can be the *SPEDIS* function and the *SOUNDEX* function. All these functions are well documented online, some really useful explanations of the methodology that stands behind their calculation can be found in multiple papers and SAS blogs.



At each iteration of the loop on Verbatim Terms, two dictionary datasets will be created in the Work Library ready to be reviewed by the Data Manager as reported in the image above. One is sorted in alphabetic order, one with the stemming algorithm proposed in this subchapter to be used when the indexes seems to perform well but 60 observations are not enough to identify the correct Drug Name. The stemming algorithm seems to find the correct drug name in most of the cases in the first 60 observations (31 cases on a total of 33 with this medications dataset).

INTERACT WITH THE USER

Now the most interesting part of the paper: the interactive window where the user can easily choose the correct matching without consulting any huge dictionary or searching text on google trying to find the real name of a drug misspelled on the CRF. Below you can see what the output window will look like. In the original raw dataset of 200 observations, only 33 of them without duplicates do not perfectly match with the dictionary. The screenshot attached below is an example related to the 4th observation of our dummy concomitant medications raw dataset, where the investigator inserted the text "BECLAMETASONE" taken for COPD by INHALATION. This observation didn't find any correct match on the dictionary.

```
/-------------------------------------------------------------------------------------------------\
|------------------------------- STEP 1: Select the direct match --------------------------------|
|-------------------------------------------------------------------------------------------------|
| DRUG (Verbatim): BECLAMETASONE                                                                  |
| ROUTE: RESPIRATORY (INHALATION)                                                                 |
| INDIC: COPD                              STEP1 - COMPLETED: 4 / 33                              |
\-------------------------------------------------------------------------------------------------/

 1   BECLOMETASONE                                  31   FLUMETASONE
 2   BECLOMETASON                                   32   HALOMETASONE
 3   BECLOMETASONA                                  33   MEDIAMETHASONE
 4   BECLOMETHASONE                                 34   PARAMETASONE
 5   BETAMETASON                                    35   ZAMETASONE
 6   BECLOMETASON CF                                36   CLOBETASONE
 7   BECLOMETASON GF                                37   BECLOMETASON NEVEL
 8   BECLOMETASONUM                                 38   BECLOMETHASONE COX
 9   BETAMETASONA                                   39   BETAMETHASONE EG
10   BETAMETASONE N                                 40   BETAMETASONE DOC
11   BETAMETHASONE                                  41   BETAMETASONE LFM
12   ALCLOMETASONE                                  42   BETAMETHASONE SA
13   DELLAMETHASONE                                 43   BECLOMET AQUA
14   DESAMETASONE                                   44   BEMETSON
15   BETAMETASONE EG                                45   BETAMETAGEN
16   BECLOMETASON-CT                                46   BUCLAMASE
17   BECLOMETASONA MK                               47   BETAMESON N
18   BECLOMETATOP                                   48   BETAMETAGEN N
19   BETAMETHASON                                   49   BECLAZONE
20   BETAMETHASONE G                                50   BETAMETASONA MK
21   BETAMETHASONE N                                51   BETAMETHASON GF
22   BECLASONE                                      52   BETAMETHASONUM
23   BETAMETASOMA                                   53   BATASONE
24   BUCOMETASONA                                   54   BECADEXAMINE
25   DEXAMETASON                                    55   BECASONE
26   ALCLOMETASONA                                  56   BECATECARINE
27   DERMETASONE                                    57   BECLOMET FORT
28   DEXAMETASONA                                   58   BECLOSONE
29   DEXAMETHASONE                                  59   BESTASONE
30   ENDAMETHASONE                                  60   BETAMETHAZINE

Write the number of the correct direct match or enter search followed by new term to search
You can also check in coding.DRUG_CHECK_STEM or DRUG_CHECK_AZ and write the direct match here.
Hit enter leaving this space blank to ignore this drug. Write SAVEANDEXIT to save and exit.
_____________________________________________________________________________
```

The process works like this: start by creating a shell with macro variables that will stand totally outside from the procedure, give a value to these macro variables, and finally prompting the window, obtaining the value inserted in the underlined field at the bottom of the page with another macro variables.

WINDOW SHELL BUILDING

For completing the tool, three different window shells will be created, the first one will allow for the matching of the Verbatim Term with the Preferred term, the second one for the matching between Preferred Term and the Substance Name, finally the third one for the matching between the Substance Name and the all-levels ATC Code. The procedure to build them is always the same, hence I will dedicate only one subchapter for the first one.
The macro statement that stands behind this output is *%WINDOW* but it should be enclosed inside another macro to be called in each iteration.
Windows are really easy to be built, with the symbol # we identify the line where that text should appear while with the symbol @ we identify at which "column" the text should start. The first rows of codes will look like this:

```
%macro finestrefrs();
%window frsstep

#2 @5 "/-------------------------------------------------------------------------------------------\"
#3 @5 "|------------------------- STEP 1: Select the direct match ---------------------------|"
#4 @5 "|-------------------------------------------------------------------------------------------|"
#5 @5 "| DRUG (Verbatim): &valuez."
#5 @103 "|"
#6 @5 "| ROUTE: &routez."
#6 @103 "|"
#7 @5 "| INDIC: &cmindcz."
#7 @45 "  STEP1 - COMPLETED: &k. / %cmpres(&nobs.)"
#7 @103 "|"
#8 @5 "\-------------------------------------------------------------------------------------------/"
#10 @2  " 1"
#10 @7 "&drug1"
#10 @70 "31"
#10 @75 "&drug31"
```

For example in the first three rows of code nothing strange is required to be programmed by writing down the fixed text and it will be prompted as specified in the 2nd line of the window starting from the 5th column. But in the 4th row of the code related to the window, that will prompt text at line 5 we have a portion of fixed code `DRUG (Verbatim):` as well as a dynamic text prompted with the `&valuez.` macro variable. For this reason if we want to close the box we need to add the row of fixed code at the same line but at the final column.

Same for the list of records, for each row there are 2 sequential identifiers that will be reported as fixed text while the relative verbatim terms will be prompted with a macro variable as dynamic text. An interesting feature is the counter that will show the progress of the cleaning phase, obtained showing the actual iterator of the *%DO* loop and the total number of observations, end of the loop. The most important line in the output window is the last one, the underlined row that will allow the user to insert some text to interact with the process, this line can be prompted with the following code:

```
#45 @5 ncoded 80 attr=underline;
```

Basically this means that at the 45th row there will be an underlined field, and the text wrote by the user will be stored in the macro variable `&ncoded`.

It is worth mentioning the dimension of the output window as it seems that it changes depending on the screen resolution, so I had to do some adjustments to better fit both the desktop screen and the laptop screens. In some cases on laptops we had a portion of the screen cut at the top and bottom and it made the coding activity impossible while on desktops it was displayed correctly. Also remember that *%WINDOW* macro statement needs to be closed with *%END* as well as the macro call with its *%MEND* and the window will be recalled with the statement *%DISPLAY*.

OBSERVATIONS INTO MACRO VARIABLES TO FULLFILL THE WINDOW

As described in the previous subchapter, to fulfill the window with the texts needed at each iteration of the *%DO* loop on Verbatim Terms, we need to give values to macro variables. We will need both the values that identifies the Verbatim Term that we are treating (this was previously done with the DATA _NULL_ step in the previous subchapter describing the loop) and the first 90 nearest terms of the dictionary sorted with the stemming indexes. I implemented this in the tool by using a combination of *%syscall set()* and looping 60 times the function *%sysfunc(fetchobs())*. Basically, this iteration will simulate a data step, first of all opening the dataset and at each iteration reading an observation, but without feeding the PDV and giving us the opportunity to put multiple observations into different macro variables. Finally, when the end of the loop will be reached, the dataset will be closed. This can't be done inside a single DATA step using the *call symput()* function. An interesting side of the *%syscall set()* is that its parameter must be a macro variable, but it should be referred without using the ampersand symbol.

```
%let fetcher=%sysfunc(open(chem_t2,i));
%syscall set(fetcher);
%do i=1 %to 60;
        %let drug&i= ;
        %let drugf&i.= ;
        %let fetching= %sysfunc(fetchobs(&fetcher,&i));
        %let drug&i. = %qtrim(%nrbquote(&drugt.));
        %let drugf&i. = %nrbquote(&direct_match.);
%end;
%let fetcher=%sysfunc(close(&fetcher));
```

In this way we will have a set of macro variables &drug1 &drug2 … &drug60, filled with the values of the first 60 observations of the dictionary dataset, ready to fulfill the window, recalled in the next step with the macro where the *%WINDOW* macro statement is stored. This combination of macro function is not widely used in SAS macro programming but it resulted in considerable time saving for the overall process. Sometimes the decision to undertake time saving strategies rather than the usual not optimized code can seem pedantic but in this case, halving the computational time of a single iteration will result in a great amount of time saved especially on those studies where concomitant medications datasets are huge and the coding activity requires days.

HOW THE USER CAN INTERACT WITH THE WINDOW

The macro variable &ncoded, the text written in the underlined field of the window, will lead the subsequent action to be taken inside the loop, below the code containing all the possible requests by the user:

```
/* if number has been entered, this direct match will be in drug coded */
%if %datatyp(&ncoded.)=NUMERIC and %length(&ncoded.) ne 0 and &ncoded >= 1 and &ncoded <=
60 %then %do;
        drug_coded="&&drugf&ncodedc";
%end;
%else %if %datatyp(&ncoded.)=NUMERIC and %length(&ncoded.) ne 0 and (&ncoded < 1 or
&ncoded > 60) %then %do;
        drug_coded=upcase(left(trim("£NOT_CODED£")));
%end;
/* if want to quit at this stage go to save and exit section below */
%else %if %upcase(%sysfunc(compress("&ncoded."))) = "SAVEANDEXIT" %then  %do;
        %goto SAVEANDEXIT;
%end;
/* if want to search for another term go to search section below    */
%else %if %scan(%upcase(&ncodedc),1) = SEARCH %then  %do;
        %goto SEARCH;
%end;
/* direct match term has been entered */
%else %if %length(&ncoded.) ne 0 %then %do;
        drug_coded=upcase(left(trim("&ncodedc.")));
%end;
/* drug has been skipped */
%else %if %length(&ncoded.) = 0 %then  %do;
        drug_coded=upcase(left(trim("£NOT_CODED£")));
%end;
```

With the *%datatyp* macro function we can identify the nature of a macro variable, in this case we want to distinguish those cases where the value of &ncoded is numeric (related to an observation found in the window) or character for other actions.

The possible actions defined on the &ncoded macro variable are the following:

1) If the user inserted a valid numeric value inside the underlined field, this will be decoded with the relative position in the dictionary dataset. As an example related to the screenshot below, the investigator wrote in the Verbatim Text of the CRF, “DIPROBASE CREAM”, while for coding purpose only the text “DIPROBASE” will be used as Drug Name. In this case, the Data Manager will just need to write “1” in the underlined field, that is the number relative to the correct text.
2) If a not valid numeric value will be inserted, an error placeholder common for all the steps will be used in the Drug Name field: “£NOT_CODED£”. As described in the previous chapter related to the preparation of the starting dataset, this coding tool will try to code these cases again in next coding steps.
3) If the user want to interrupt the coding for any reason, they can save and exit from the process flow just writing SAVEANDEXIT, the *%GOTO* macro statement will lead to the end of the macro where the final coded dataset will be created.
4) The stemming algorithm implemented seems to perform very well where the typos or other inconsistencies are found in the middle or at the end of the string, while its guessing power seems to drop down when some useless text is wrote at the beginning of the string. For this reason if the Data Manager can visually identify the name of the drug hidden inside the string like in this case “2% w/v eye drops of Allercrom”, they can simply write the text “SEARCH ALLERCROM”. In this case another window related to the same counter will be displayed showing the best coding choices for “ALLECROM”.
5) If the Data Manager is sure for a certain drug name as for “ALLECROM” in the example proposed at point 4, the term can simply be written in the underlined field and it will be used as Drug Name once trimmed and upcased. This option is useful also when it is necessary to consult the internal dictionary created or maybe the online version of WHO-DDE on the Uppsala Monitoring Center website. When the correct Drug Name is identified it can be copy-pasted inside the window.
6) The Data Manager can decide to skip a specific Verbatim Term waiting for clarifications from the sponsor or the results of an open query, this option is available just hitting enter in the window leaving as blank the

underlined field. The term will be associated with the previously discussed placeholder "£NOT_CODED£" waiting for next steps of coding.

```
/-------------------------------------------------------------------------------------------------\
|------------------------------ STEP 1: Select the direct match -------------------------------|
|-------------------------------------------------------------------------------------------------|
| DRUG (Verbatim): DIPROBASE CREAM                                                                |
| ROUTE: TOPICAL                                                                                  |
| INDIC: dermatitis                        STEP1 - COMPLETED: 12 / 33                             |
\-------------------------------------------------------------------------------------------------/

 1   DIPROBASE                                          31   DIPROSEPT
 2   DIPROBETA                                          32   DIPROVATE MF
 3   DIPROVATE-RD                                       33   DIPROVATE-G
 4   DIPRODERM                                          34   DIPROVATE-N
 5   DIPRODERMA                                         35   DIROX ALER
 6   DIPROLENE AF                                       36   CIPROBAY HC
 7   DERMACREAM                                         37   DIPROSAC XL
 8   DIPRODERMA C                                       38   DIPROSONE G
 9   DITHROCREAM                                        39   DIPROSONE Y
10   DURACREAM                                          40   CIPROACE
11   DIPRO AS                                           41   CIPROBACH
12   DIPROBATH                                          42   CIPROBACT
13   DIPROFAST                                          43   DIPROFORM
14   DIPROVATE                                          44   DIPROSONA
15   DIPROBET                                           45   DIPROSPAN
16   DIPROLAR                                           46   PROBASE
17   DIPROSAL                                           47   CIPROCAM
18   DIPROSAN                                           48   DEPRERAM
19   DIPROSALIC                                         49   DIASERA
20   DIPROSEN                                           50   DIPROSTENE
21   DIRASEC                                            51   CIPROBAY XR
22   DIPROLENE                                          52   CIPROSER
23   DIPROSONE                                          53   DIPROSONE VO
24   DIPROTENE                                          54   DIPROGENTA
25   DIUROLASA                                          55   CIPROBETA
26   LIPOBASE                                           56   DISTOBRAM
27   CIPROBAC                                           57   CIPROREM
28   DIPROCEL                                           58   DIASERA L
29   DIPROSTA                                           59   DIPRAFENONA
30   DIROSEA                                            60   DIPRASORIN

Write the number of the correct direct match or enter search followed by new term to search
You can also check in coding.DRUG_CHECK_STEM or DRUG_CHECK_AZ and write the direct match here.
Hit enter leaving this space blank to ignore this drug. Write SAVEANDEXIT to save and exit.
1
```

PHASE 2: DRUG NAME TO PREFERRED NAME TO ATC CODES

Phase 2 is much simpler than the previous one, it is divided into 2 different steps, neither of them mandatory, and prompted only when a specific Drug Name can be associated with more than one Preferred Name or when a specific Preferred Name can be associated with more than one ATC all-level code. In these cases, an interaction with the Data manager is required, otherwise when the association is unique, the tool will automatically match the correct text.

The MED dictionary dataset will be filtered only on those Drug Names actually resulting from phase 1, this important dimension reduction will reduce the execution time of this huge dataset in a very significant way. A *%DO* loop will repeat on the complete dataset without duplicates and without errors and skipped terms (both identified with the placeholder "£NOT_CODED£"). At each iteration the process will work the way described below, on each single observation.

First of all I need to merge the dataset resulting from phase 1 with the dictionary containing the matching between Drug Names and Preferred Names, only on those terms with a unique match. If after this merge the specific observation on which the tool is working on will have a value, than we will skip the interaction of the first step. Otherwise if it will still be blank than the interactive window need to be displayed showing all the possible Preferred Names. Most of the time, only two Preferred Names are reported as in the example below, this is due to the same Drug Name marketed in different countries from different companies with various compositions. A dataset stored in the Work Library will contain the same Terms reported in the window, with an additional variable reporting all the ingredients of the specific Preferred name, this is a crucial information needed for the decision. In the next version of this tool actually under development, only 30 possible options will be shown, and the remaining space will be filled with the specification of the ingredients of these Preferred Names.

```
/-------------------------------------------------------------------------------------------------\
|------------------------------ STEP 2: Select the Preferred Term ------------------------------|
|-------------------------------------------------------------------------------------------------|
| DRUG (direct match): ALLERCROM                                                                  |
| ROUTE: OPHTHALMIC                                                                               |
| INDIC: Hayfever                          STEP2 - COMPLETED: 3 / 163                             |
\-------------------------------------------------------------------------------------------------/

 1   CROMOGLICATE SODIUM                   31
 2   CROMOGLICIC ACID                      32
 3                                         33
 4                                         34
_________________________________________________________________________________________________

29                                         59
30                                         60

   Can you see listed the correct Preferred Term? If yes, please write the number.
   You can find more information for these preferred terms in coding.PT_CHECK.
   Hit enter leaving this space blank to ignore this drug. Write SAVEANDEXIT to save and exit.
   ____________________________________________________________
```

Once completed the first step, both if automatically or with the user interaction, another merge will be done with the dataset containing the matching between Preferred Names and all the possible ATC all level coding. Most of the time an interaction is not required but, in some cases, the same substance can be used for different pathologies or even in different forms, here the indication, route and, when included, the form of the medication are crucial for the coding decision.

```
/-------------------------------------------------------------------------------------------------\
|------------------------------ STEP 3: Choose the correct ATC ------------------------------|
|-------------------------------------------------------------------------------------------------|
| DRUG (direct match): ALLERCROM                                                                  |
| PREFERRED NAME: CROMOGLICATE SODIUM                                                             |
| ROUTE: OPHTHALMIC                                                                               |
| INDIC: Hayfever                                                                                 |
|                                      STEP3 - COMPLETED: 3 / 163                                 |
\-------------------------------------------------------------------------------------------------/

1  LEVEL:4   ATC-Code: A07EB
   A : ALIMENTARY TRACT AND METABOLISM ; A07 : ANTIDIARRHEALS, INTESTINAL ANTIINFLAMMATORY/ANTIIN
   A07E : INTESTINAL ANTIINFLAMMATORY AGENTS ; A07EB : ANTIALLERGIC AGENTS, EXCL. CORTICOSTEROIDS

2  LEVEL:4   ATC-Code: D11AH
   D : DERMATOLOGICALS ; D11 : OTHER DERMATOLOGICAL PREPARATIONS
   D11A : OTHER DERMATOLOGICAL PREPARATIONS ; D11AH : AGENTS FOR DERMATITIS, EXCLUDING CORTICOSTEROIDS

3  LEVEL:4   ATC-Code: R01AC
   R : RESPIRATORY SYSTEM ; R01 : NASAL PREPARATIONS
   R01A : DECONGESTANTS AND OTHER NASAL PREPARATIONS FOR TOP ; R01AC : ANTIALLERGIC AGENTS, EXCL. CORTICOSTEROIDS

4  LEVEL:4   ATC-Code: R03BC
   R : RESPIRATORY SYSTEM ; R03 : DRUGS FOR OBSTRUCTIVE AIRWAY DISEASES
   R03B : OTHER DRUGS FOR OBSTRUCTIVE AIRWAY DISEASES, INHAL ; R03BC : ANTIALLERGIC AGENTS, EXCL. CORTICOSTEROIDS

5  LEVEL:4   ATC-Code: S01GX
   S : SENSORY ORGANS ; S01 : OPHTHALMOLOGICALS
   S01G : DECONGESTANTS AND ANTIALLERGICS ; S01GX : OTHER ANTIALLERGICS

6


7



   Can you see here the real atc code? If yes, please write the number.
   Else check in coding.ATC_CHECK and write here the ATC-CODE.
   Hit enter leaving blank this space to ignore this atc-code. Write SAVEANDEXIT to save and exit.
   ____________________________________________________________
```

Any possible interaction with the window presented in the previous chapter will still be present in these phases, anyway some of them are almost useless. The only interactions actually widely used in these phases are the number relative to the observation in the window, the skip option, the save and exit option, and only in few cases the copy paste when more than 7 possible ATC all-level codes can be associated with the Preferred Name.

After this second step, the coding for the actual record will be considered as final and the loop will start phase 2 again with another observation. After all the observations are managed, this coding phase will be finalized.

FUTURE DEVELOPMENT

This tool was written as an interesting challenge far from standard SAS activities. For this reason I feel like it is really personal and many future development possibilities came to my mind, but they are almost all related to my personal interest rather than on the intended usage.

One challenge could be to port this procedure to the SAS/AF package building a real GUI rather than the output window used in this paper. This would lead to an end-user application in closed format giving to this procedure the aspect of a real software. This would also increase its value on the programming side, make it easier to be lunched even without any SAS knowledge and prevent the code to be accidentally modified.

Some interesting implementations could be associated with the improvement of the guessing of real Drug Names both on the algorithm and on the data. On the algorithm side, a lot of literature is present on the web about stemming methodologies because it is an important branch of linguistic morphology for the development of artificial intelligences, a hot topic for today. This is the reason why stemming algorithms are mostly associated with machine learning methodologies like neural networks. On the data side, a score can be implemented directly on the dictionary to identify those Drug Names more common than the others. The Medications collected in the CRF can usually be divided into two different categories; the drugs related with the study treatment or disease and the common drugs very likely to be present in each study with common pathologies (blood pressure, cholesterol and so on). For the comment drugs the score can be implemented and it can interact with the ordering of the Drug Names to be displayed in the window, since they are much more likely to be chosen by the Data Manager as the solution.

SAS is the golden standard for clinical trials, but the coding activity is not related with the statistical analysis and for this reason it could be performed with any other environment able to lead to correct results and manage SAS datasets. I started porting this tool to Python with very good results in term of computational time.

CONCLUSION

This tool can handle all the phases of coding the medications using the WHO-DDE. It resulted in a great time saving and simplification for Data Managers. All the possible automation of the process were implemented asking for a user interaction only when strictly necessary. A guessing algorithm was implemented where the Verbatim Term, usually collected in the CRF as a free text field, needed to be matched with a Drug Name on the dictionary. The SAS output window was used to display all the information needed where the coding process required an interaction. This tool is validated and working well but it is still under development and maintenance to improve the process and keep it updated with the new dictionary releases.

ACKNOWLEDGMENTS

Special thanks to:
PM for introducing me to a world quickly became passion
MF the next paper will have two authors
MZ for the great DM support
CT and MW for the accurate review
TH and DAM.

CONTACT INFORMATION

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

Gianluca Mortari
CROS NT srl
Via Germania, 2
Verona, 37136 (ITALY)
Work Phone: +39 045 820 26 66
Fax: +39 045 820 58 75
Email: gianluca.mortari@crosnt.com
Web: crosnt.com