

SAS® Match Merges: The Driving Force behind a Web-Based Data Coding System

Barbara Felts, Wafa Handley, Chuchun Chien, Inga Allred
RTI International, Research Triangle Park, NC

ABSTRACT

Surveys often use open-ended questions to collect information that are not specifically covered in the questionnaire. These responses need to be numerically coded for analysis. The potential exists for a high volume of responses that will require coding. Consequently, there was a need to develop a system that would automatically utilize responses that had been previously processed in order to reduce the coder burden.

We have developed a system to standardize the coding of the data online. This system relies on SAS® Match Merges and Macros to build and use data dictionaries. It also allows for multiple analysts to code or review data throughout the collection period, thereby reducing the turnaround time.

This paper will outline the major processes involved in this system including v8.2 SAS® Macros, SAS® Match Merge and SAS/Access to ODBC drivers to store the data in SQL tables which are accessed by the Web-Based Data Coding System.

INTRODUCTION

In SAS, the Merge statement is used to combine observations from two or more data sets. It combines variables in the PDV (program data vector) from each observation in the input data sets to construct one observation for the output data set. When a BY statement is used for the Merge, the data sets must be sorted or indexed on the order of the matching variable or variables in the BY statement. In a Match Merge, observations are matched by the sort or indexed variables before the values are combined. Each input data set must contain the BY variable. Our process uses a unique single variable in the data sets for the BY statement in the Match Merge. We create an individual data set for each alphanumeric variable that has to be coded. We then match merge each of these data sets with the existing dictionary by renaming the string variable of the data dictionary to the name of the variable in the data set to accomplish the Match Merge.

BACKGROUND

Our survey collects data through Audio Computer Assisted Self Interviewing (ACASI) using laptops. The questionnaire relies upon open ended questions to collect information on a variety of items including names and types of drugs or tobacco products used by respondents. We receive 2,000 to 4,000 such responses per year, creating a need for quick and efficient processing. The data are transmitted from the laptops to RTI on a daily basis. They are then processed into a SAS data set and made available for use.

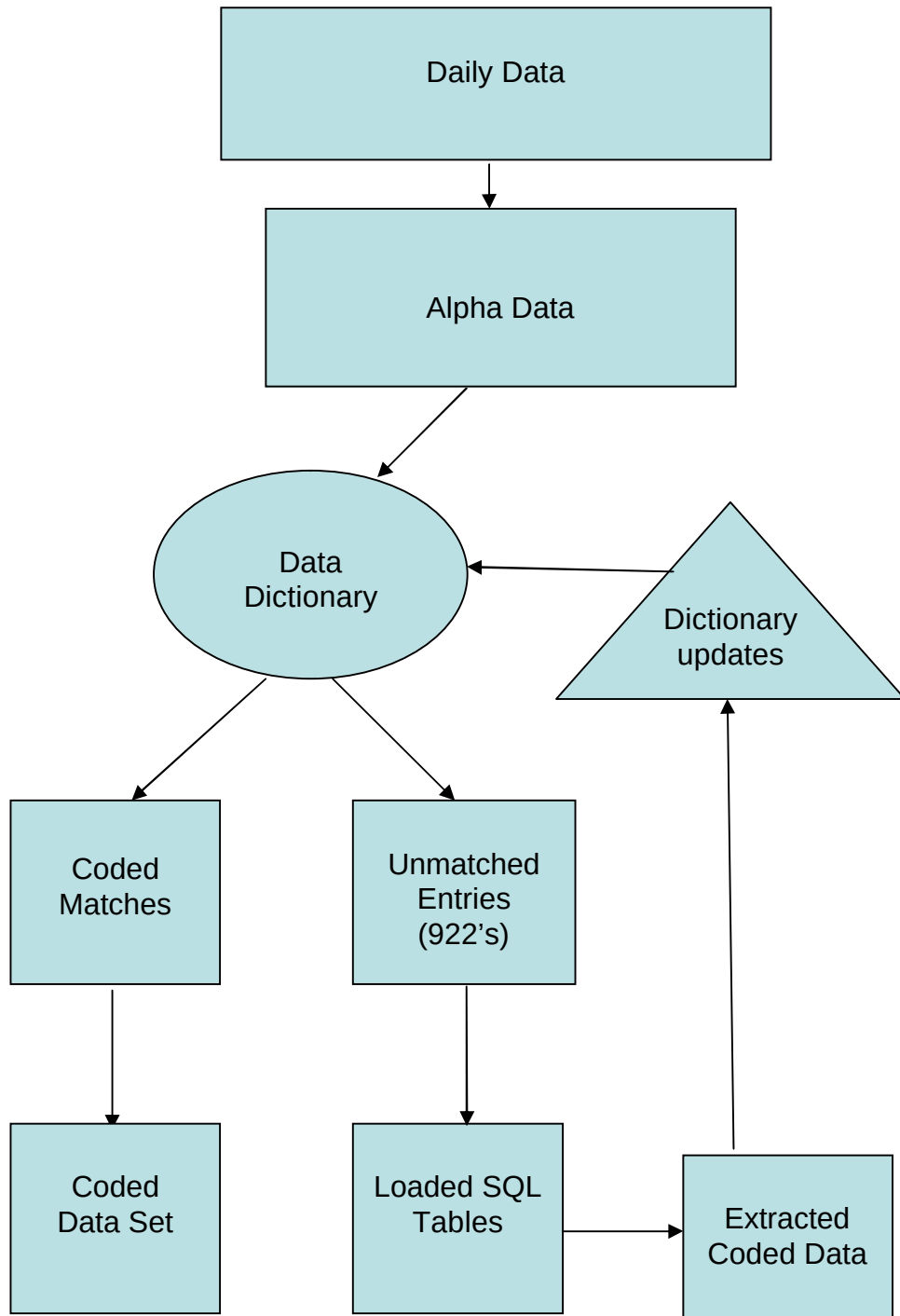
PROCESS OVERVIEW

The term "Data Dictionary" as used in this paper describes a SAS data set with two variables, one containing the string which is entered by the respondent and the other containing a numeric code assigned to that string.

The daily processed SAS data set is subset for all data where alpha responses were entered. These data are then run through the existing data dictionaries. Through SAS® Match Merge statements, matching entries are coded and unmatched entries are identified.

Currently, the system handles over 60 variables that need to be coded. We use SAS® Macros to create an individual data set for each variable, merge each of the data sets with the existing data dictionary, using the string variable as the BY variable. If an exact match occurs, the corresponding code is assigned. If no match is found, a code of 922 is assigned. Visual Basic (VB) is used to populate SQL tables with the resulting data for use in the Web-Based Coding system. SAS/Access to ODBC is used to interface with SQL's ODBC driver through SQL passthrough to extract coded data and update data dictionaries.

PROCESS OVERVIEW



PROGRAM STEPS

1. Create a macro to create individual data sets and Match Merge each data set with the data dictionary.
2. Set a macro variable containing the parameters to be passed to the macro.
3. Loop through the macro variable to create all data sets.
4. Sort the data sets.
5. Create final data sets containing all the variables.
6. Write data to a text file for loading into the web-based data coding system.
7. Use Visual Basic to load the data into the web system for coding.
8. Use Proc SQL to extract the coded data and update the data dictionaries.

STEP 1. CREATE A MACRO TO CREATE INDIVIDUAL DATA SETS AND MATCH MERGE EACH DATA SET THE DATA DICTIONARY.

```
OPTIONS SYMBOLGEN SOURCE2 ;

libname rawdata "\\xxxxxx\xxx\xxxdev\2005\postdata\ssds";
libname webdsn "\\xxxx\xxx\edit\xxx2005\xxxsp\dictionaries\ssds";

%let indsn = rawdata.daily;
%let drugdsn = webdsn.drugspd;
%let drgfinal = webdsn.dgfinald;
%let cumpdsn = webdsn.cumprev;
%let cumfdsn = webdsn.cumfinal;

/* subset raw data for the sp vars*/

%macro createdata (spscurr, spprev, spdaily, spdict, spfinal, spvar, scvar);

/* create the individual data (&spscurr) from the raw data set (&drugdsn) */
data &spscurr ;
    set &drugdsn;
    &spvar=upcase(&spvar);
    if &spvar ne " " then output;
run;

/*sort the data for the merge*/
proc sort data=&spscurr;
    by &spvar;
run;

/*only the new cases need to go into &spdaily since ones in &spprev have
already been loaded to the web interface already*/
data &spdaily
    webdsn.&spprev;
    merge webdsn.&spprev(in=prev)
          &spscurr (in=curr)
    ;by &spvar;
    output webdsn.&spprev;
    if curr /*and not prev*/ then do; output &spdaily; end;
run;

/*Rename the variable spstring in the drug dictionary to spvar enable the
merge to work properly */
data &spdict;
    set webdsn.drugdict(rename= (spstring = &spvar));
    output;
run;

/*Sort the dictionary dataset &spdict by the spvar*/
proc sort data = &spdict;
    by &spvar;
```


STEP 3. LOOP THROUGH THE MACRO VARIABLE TO CREATE ALL DATA SETS.

```

%let i = 1;
%macro alldata;
  /*Loop through each line in the variable allvarsdata*/
  %do %while (%scan (&allvarsdata,&i) ne);
    %let datasetvar = %scan (&allvarsdata,&i);
    %let prevvar = %scan (&allvarsdata,%eval(&i+1));
    %let dailyvar = %scan (&allvarsdata,%eval(&i+2));
    %let dictvar = %scan (&allvarsdata,%eval(&i+3));
    %let finalvar = %scan (&allvarsdata,%eval(&i+4));
    %let spvar = %scan (&allvarsdata,%eval(&i+5));
    %let scvar = %scan (&allvarsdata,%eval(&i+6));
  /*calls the macro createdata for each line in allvarsdata*/
  %createdata(&datasetvar, &prevvar, &dailyvar, &dictvar, &finalvar, &spvar,
    &scvar);
  %let i=%eval(&i+7);
%end;
%mend;

```

STEP 4. SORT THE DATA SETS.

```

/*create a macro called sortdata that will sort the data sets by questid*/
%macro sortdata (unsortpv, sortpv, sortdly, sortfnl);
proc sort data = webdsn.&unsortpv out = &sortpv;
  by questid;
run;
proc sort data = &sortdly out = &sortdly;
  by questid;
run;
proc sort data = &sortfnl out = &sortfnl;
  by questid;
run;
%mend;

/*create a macro variable with the related data sets that were created
earlier*/
%let sortdatavars=
  ls1prev ls1temp ls1daily ls1final
  ls2prev ls2temp ls2daily ls2final
  ls3prev ls3temp ls3daily ls3final
  ls4prev ls4temp ls4daily ls4final
  ls5prev ls5temp ls5daily ls5final
  -----
  -----some lines removed to conserve space-----
  -----
  tx33prev tx33temp tx33daily tx33finl
  tx34prev tx34temp tx34daily tx34finl
  tx35prev tx35temp tx35daily tx35finl;
/*Read in each line from the macro variable sortdatavars and
pass the variables to the macro sortdata*/
%let i = 1;
%macro allsortdata;
%do %while (%scan (&sortdatavars,&i) ne);
  %let unsortpv = %scan (&sortdatavars,&i);
  %let sortpv = %scan (&sortdatavars,%eval(&i+1));
  %let sortdly = %scan (&sortdatavars,%eval(&i+2));
  %let sortfnl = %scan (&sortdatavars,%eval(&i+3));
  %sortdata (&unsortpv, &sortpv, &sortdly, &sortfnl);
  %let i=%eval(&i+4);
%end;
%mend;
%allsortdata;

```

STEP 5. CREATE FINAL DATA SETS CONTAINING ALL THE VARIABLES.

```

/* create cumulative previous dsns */
data &cumpdsn;
  merge ls1temp ls2temp ls3temp ls4temp ls5temp
        in1temp in2temp in3temp in4temp in5temp
        pr1temp pr2temp pr3temp pr4temp pr5temp
        tr1temp tr2temp tr3temp tr4temp tr5temp
        st1temp st2temp st3temp st4temp st5temp
        sv1temp sv2temp sv3temp sv4temp sv5temp
        sd1temp sd2temp sd3temp sd4temp sd5temp
        tx21temp tx22temp tx23temp tx24temp tx25temp
        tx31temp tx32temp tx33temp tx34temp tx35temp
  ;by questid;
  output;
run;
proc sort data = &cumpdsn nodupkey;
  by questid;
run;

/* create final dsn */
data &drgfinal;
  length LS01SC1-LS01SC5 IN01SC1-IN01SC5 PR05SC1-PR05SC5
        TR05SC1-TR05SC5 ST05SC1-ST05SC5 SV05SC1-SV05SC5
        SD05SC1-SD05SC5 TX21SC1-TX21SC5 TX36SC1-TX36SC5 4.;
  Merge ls1final ls2final ls3final ls4final ls5final
        in1final in2final in3final in4final in5final
        pr1final pr2final pr3final pr4final pr5final
        tr1final tr2final tr3final tr4final tr5final
        st1final st2final st3final st4final st5final
        sv1final sv2final sv3final sv4final sv5final
        sd1final sd2final sd3final sd4final sd5final
        tx21finl tx22finl tx23finl tx24finl tx25finl
        tx31finl tx32finl tx33finl tx34finl tx35finl;
  by questid;
  array drugsc{*} LS01SC1-LS01SC5 IN01SC1-IN01SC5 PR05SC1-PR05SC5 TR05SC1-
  TR05SC5
                  ST05SC1-ST05SC5 SV05SC1-SV05SC5 SD05SC1-SD05SC5 TX21SC1-
  TX21SC5
                  TX36SC1-TX36SC5;
  do i = 1 to dim(drugsc);
    if drugsc{i} = . then drugsc{i} = 9998;
  end;
  output;
run;

proc sort data = &drgfinal nodupkey;
  by questid;
run;

/* get back the raw vars in their original case*/
data &drgfinal;
  merge &drgfinal(in=final keep = questid
        ls01sc1-ls01sc5 IN01SC1-IN01SC5 PR05SC1-PR05SC5
        TR05SC1-TR05SC5 ST05SC1-ST05SC5 SV05SC1-SV05SC5
        SD05SC1-SD05SC5 TX21SC1-TX21SC5 TX36SC1-TX36SC5)
        &drugdsn (in=drdsn keep = questid Tbegintr fiid LS01HS1-LS01HS5
        IN01OTH1-IN01OTH5 PR05A PR05B PR05C PR05D PR05E TR05A TR05B
        TR05C TR05D TR05E ST05A ST05B ST05C ST05D ST05E SV05A SV05B
        SV05C SV05D SV05E SD05A SD05B SD05C SD05D SD05E
        TX21SP1-TX21SP5 TX36SP1-TX36SP5);
  by questid;
  if drdsn;
  output;
run;

```

STEP 6. WRITE DATA TO A TEXT FILE FOR LOADING INTO THE WEB-BASED DATA CODING SYSTEM.

```

libname webdsn "\\xxxx\xxx\edit\xxx2005\ dictionaries\ssds";
filename inputd "\\xxxx\xxx\edit\xxx2005\ dictionaries\inpgms\drugind.txt";
%let indsn1 = webdsn.drgfinal; /* This data set was created in step 5 */
%let indsn2 = webdsn.prevd922; /* A copy of yesterday's cumd922 */
%let outdsn = webdsn.cumd922; /* A cumulative Dataset with already processed
ids*/
/*Because we could have a case come in more than once, we must take the dataset
that was output from Step 5 "dgfinald" and check it against the records that have
come in before. The text file should only include the cases that have not been
processed previously.*/

data &outdsn (keep=QUESTID SPFYCODE drugvar DROP=K1-K5);
  RETAIN K1-K5 0.;
  Merge &indsn1 (IN=F1 keep=questid <var-list>)
        &indsn2 (IN=F2) END=LASTREC ;
  BY QUESTID;
  If F2 then codestat = 1;  else codestat = 0;

  IF F1 and F2 and spfycode=1 THEN
    DO; K1=K1+1; END;
  ELSE IF F2=1 AND F1 NE 1 THEN
    DO; K2=K2+1; END; /*This one has been processed so leave it alone.*/
  ELSE DO; K3=K3+1; /*Set spfycode=0 If F1 and ^F2 so it will be processed.*/
    SPFYCODE=0; END;

  /*If the case is in the final data set (dgfinald) but not in the previously
  processed data, then process it.*/
  IF SPFYCODE=0 THEN
    DO;
      K4=K4+1;
      Array drugsps{*} PR05A <var-list for text vars>;
      Array drugscs{*} PR05SC1 <var-list for matching code vars> ;
      drugvar=1;
      SC1 = 0;
      SC2 = 0;
      /*Output to a text file and a dataset*/
      file inputd lrecl=365;
      do i = 1 to dim(drugscs);
        if drugscs{I} eq 922 then
          do;
            questvar=" ";
            call vname (drugscs{I},questvar);
            codestat='0';
            put @1 drugsps{I} $50. @51 questid $7. @62 questvar $8. @74 codestat 1.
              @77 SC1 @83 SC2 @89 IN01SC1 4. @95 IN01SC2 4. @101 IN01SC3 4.
              @107 IN01SC4 4. @113 IN01SC5 4. @119 LS01SC1 4. @125 LS01SC2 4.
              @131 LS01SC3 4. @137 LS01SC4 4. @143 LS01SC5 4. @149 PR05SC1 4.
              @155 PR05SC2 4. @161 PR05SC3 4. @167 PR05SC4 4. @173 PR05SC5 4.
              @179 ST05SC1 4. @185 ST05SC2 4. @191 ST05SC3 4. @197 ST05SC4 4.
              @203 ST05SC5 4. @209 SV05SC1 4. @215 SV05SC2 4. @221 SV05SC3 4.
              @227 SV05SC4 4. @233 SV05SC5 4. @239 TR05SC1 4. @245 TR05SC2 4.
              @251 TR05SC3 4. @257 TR05SC4 4. @263 TR05SC5 4. @269 SD05SC1 4.
              @275 SD05SC2 4. @281 SD05SC3 4. @287 SD05SC4 4. @293 SD05SC5 4.
              @299 TX21SC1 4. @305 TX21SC2 4. @311 TX21SC3 4. @317 TX21SC4 4.
              @323 TX21SC5 4. @329 TX36SC1 4. @335 TX36SC2 4. @341 TX36SC3 4.
              @347 TX36SC4 4. @353 TX36SC5 4.;
          end;
        end;
        SPFYCODE=1;
      END;
    END;
  
```

```
output &outdsn; /*add the record to the previously processed ones on the
cumd922*/
run;
```

STEP 7. USE VISUAL BASIC TO UPLOAD THE TEXT FILE TO THE WEB TABLE FOR CODING.

**/*Below is a sample of the data in the output text file from Step 6.
The actual data was changed to protect the survey and some of the columns were
removed to conserve space*/**

```
Alpha data      1234567 PR05SC1 1 0 0 9998 9998 ...<text removed>... 9998 0 0 0
Alpha data      7654321 LS01SC1 1 0 0 9998 9998 ...<text removed>... 9998 0 0 0
```

```
/*vb code to append the text file to a table*/
Dim dbConnection As New ADODB.Connection
Dim strdbConnection As String
Dim strSQL As String
Dim rcssample As New ADODB.Recordset
Dim intfilenum As Integer 'vb file number
Dim strfilename As String 'path and file name
Dim strinline As String 'one line of input from the ascii file
Let intfilenum = FreeFile()
Let strfilename = "drugind.txt" /*text file created in step 6 */
Open strfilename For Input As #intfilenum 'Open the text file
'Connect to the SQL Server database
strdbConnection = "driver={SQL
Server};server=**;uid=**;pwd=**;database=**;dsn='"
dbConnection.ConnectionString = strdbConnection
dbConnection.Open strdbConnection
dbConnection.Execute ("Truncate table SPCD_bkdailydrugin")
dbConnection.Execute ("Insert into SPCD_BkdailyDrugin select * from
SPCD_dailydrugin ")
'Openrecordset
rcssample.Open "Select * FROM SPCD_dailydrugin", dbConnection, adOpenDynamic,
adLockOptimistic

Do While Not EOF(intfilenum)
Line Input #intfilenum, strinline
rcssample.AddNew
rcssample!spvar = Mid(strinline, 1, 50)
rcssample!questid = Mid(strinline, 51, 7)
rcssample!questvar = Mid(strinline, 62, 8)
rcssample!codestat = Mid(strinline, 74, 1)
rcssample!SC1 = CInt(Mid(strinline, 77, 4))
rcssample!SC2 = CInt(Mid(strinline, 83, 4))
rcssample!IN01sc1 = CInt(Mid(strinline, 89, 4))
rcssample!LS01sc1 = CInt(Mid(strinline, 119, 4))
/*I removed some lines like the ones above in order to conserve space*/
rcssample!SC3 = CInt(Mid(strinline, 359, 4))
rcssample!SC4 = CInt(Mid(strinline, 365, 4))
rcssample!SC5 = CInt(Mid(strinline, 371, 4))
rcssample.Update
Loop
Close #intfilenum
rcssample.Close
dbConnection.Close
End
```

The coders can then view the data through a web-based coding system which was created with Cold Fusion. This system allows the analysts to use a web interface to enter numeric codes for the unmatched data. This system also allows them to view the current dictionary to aid them in their coding efforts. Steps 1 - 7 are executed daily. At the end of the quarter when all the coding is finished we proceed to Step 8.

STEP 8. USE PROC SQL TO EXTRACT THE CODED DATA AND UPDATE THE DATA DICTIONARIES.

```
/*Extract the data from the SQL tables*/
proc sql;
  connect to odbc
  (dsn="database" uid="username" pwd="*****");
  create table webdsn.extractdict as
  select *
  from connection to odbc
  (select sc1, spvar
  from spcd_DailyDrugIn
  where codestat = '2' and sc2 = 0)
  order by sc1;
  disconnect from odbc;
quit;

/*Make the text in spvar all uppercase*/
data webdsn.upextractdict;
  set webdsn.extractdict;
  spvar = upcase(spvar);
run;

/*Sort the extracted dictionary and remove the duplicates*/
proc sort data = webdsn.upextractdict nodupkey;
  by spvar;
run;

/*Combine the previous dictionary and the extracted one*/
data webdsn.drugdict (keep= dccode spstring);
  set webdsn.prevdictionary;
  webdsn.upextractdict(rename=(sc1=dccode spvar=spstring)) ;
  spstring=upcase(spstring);
run;
```

Once the analysts have entered all the missing codes, we extract the data and append it to the data dictionary for future use. Frequent processing adds more data to the dictionaries allowing for more automated coding by the SAS processes and less manual coding to be done by the analysts.

CONCLUSION

The use of the SAS® Macro Language and SAS® Match Merge provides the tools to create and manage code in a creative and efficient way. The SAS® Macro Language is used to create and process a multitude of data sets and is an effective tool in controlling the amount of code needed to complete the task. The SAS® Match Merge facilitates the use of data dictionaries to reduce the amount of manual work required. The web-based system for the unmatched responses allows the analysts to complete the coding throughout the quarter as they are received. As the data dictionaries are expanded, the need for manual coding will reduce even more.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact any of the authors at:

Barbara Felts	bfelts@rti.org	(919) 541-6938
Wafa Handley	handley@rti.org	(919) 541-6066
Chuchun Chien	cchien@rti.org	(919) 485-5552
Inga Allred	irb@rti.org	(919) 541-7072

Mailing address for all authors:

RTI International

P.O. Box 12194

Research Triangle Park, NC 27709-2194

Web: <http://www.rti.org>, Fax: (919) 541-6178

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.