Calling a Remote Macro Using %INCLUDE or a Stored Macro Facility

Tony Reeves, Research Analyst, Wisconsin Department of Health Services

Abstract

Many of us find SAS® macros intimidating. They often bring to mind images of doubled ampersands and percent signs spinning off in nested iterations that are intricate as the julias in a Mandelbrot image. However, macros can be quite useful. In particular, when a segment of code is likely to be broadly shared but must be routinely updated in a manner that is uniform and accurate, it can be useful to have a remote macro that many SAS® programmers can access but only one or a few can edit. This paper creates an example of such a macro. It then describes two complimentary methods of programming it so that it can be invoked by programmers from remote locations. The first uses %INCLUDE coding to identify the remote SAS® program file containing the macro. The second creates a stored macro facility to call the same macro. My goal will be to be to stress the advantages of each approach at different points in the process of developing and using such a macro.

BACKGROUND: Why Use Macros, and in Particular, One that is Remote?

In my SAS® programming, I make limited use of macros. I use macros to enter values for important, frequently changed date and directory parameters that occur across a program. I place the macros that automatically change these parameters in prominent positions at the top of the program. This allows someone unfamiliar with a periodic report I run to update and submit the program generating it without mastering all the intricacies of my coding. This in turn makes it possible for me to call in sick when I'm not feeling so good, or even to have a nice vacation occasionally. I also use macros to identify, sort and list potential data entry errors in files I am trying to clean. However, the most important use I make of macros is to take commonly-used coding and make it available to other SAS programmers in a way that is consistently and accurately edited.

This last macro use is particularly important with health care assessment databases such as the Outcomes and Assessment Information Set (OASIS), a repository of diagnostic and health history data for patients who are being reimbursed for home health agency services by Medicare or Medicaid. These medical data sources often have restricted geographical information about individual patients. In the case of OASIS, for instance, the only patient address information provided is the five digit zip code. This limitation may be intended to protect patient confidentiality. However, it also frustrates efforts to use OASIS data to compare home health care patients across the state in terms of the issues that are of concern to Wisconsin healthcare policymakers: for instance, patients' level of mental or physical function, their most common diagnosed medical conditions, their ability to engage in personal hygiene and other self-care, and their probability of having suffered recent falls and other accidents. The enormous number of zip codes in a state the size of Wisconsin makes them an inconvenient way to comparing geographically diverse home health care patients. Counties, however, are a comparatively compact, widely recognized set of political subdivisions. Counties also determine the boundaries of the administrative regions that make up the Division of Quality Assurance, the investigatory branch of the Wisconsin Department of Health. Therefore, coding that assigns zip codes to their appropriate counties is a very useful tool for statistical programming that relies on the OASIS system.

I was able to get a roster that matches zip codes and counties through the Wisconsin Department of Health Services.[1] Of course, once I downloaded this into a file that was readable by SAS®, I had to address a number of preliminary issues. The first was how to assign a zip code to a county, when the zip code straddled two or more counties. A closely related question was what to do with a zip code that showed up in a patient's OASIS assessment, but did not appear in the zip code-to-county roster. My favorite tool for both of these tasks is WI HomeTownLocator.[2] This resource will tell you the county or counties that a given zip code is in. Where a county is in more than one county, it tells which county the code is centered in geographically. This at least provides some basis for choosing one county from two or more possibilities. Since post offices are occasionally closed down, causing their zip codes to be reassigned or discontinued, the zip code-to-to-county roster must be updated periodically. My rule of thumb is to update this list at six month intervals. Typically I find no more than a dozen codes that have to be reassigned or eliminated.

Once you have a satisfactory roster of zip code-to-county matches, your most important question is how to incorporating this into your SAS® program. One option, of course, is to assign it directly to the program you are going to be submitting in SAS®. If you only plan to go against OASIS very occasionally, this might be acceptable. However, the list promises to be extremely long. My roster of Wisconsin state zip codes, for instance, fills up 916 lines, or approximately twenty-five screens. This is pretty ungainly. If the zip code-to-county conversion is to be used across time repeatedly, or in several different programs, the risk increases that various programs have data that is, to differing degrees obsolete. This risk increases even more, of course, if you share your roster code with other programmers who must also access OASIS or another system that uses only zip codes to identify people geographically. One veteran SAS® programmer has defined this problem, in a slightly different context, as "versionitis".[3] Clearly, a coding macro that is used this widely, and that is so subject to the need for periodic revision, should be stored where it can be accessed by many programmers, but can be altered only by one or a few. In the balance of this paper I will describe two different means of setting up such a system for remote, limited access. One method will involve using the %INCLUDE command to call remotely the zip code conversion macro. In the other the same thing will be accomplished by setting up a very simple stored macro facility.

OPTION 1: USING %INCLUDE TO CALL A REMOTE MACRO

Appendix 1A has an example of a %INCLUDE call incorporated in a SAS® program that I created. It generates a table that contains counts of the number of Wisconsin home health agency patients, by county and statewide, who had experienced at least one fall at some point in Calendar Year 2008. The heart of this program is contained in two lines:

```
Filename zipcode 'c:\stored_macros\macro_zipcode.sas';

%INCLUDE zipcode;
```

---

[1] It is also possible to use a number of resources to get statewide zip code information, including the online Postal Service database at http://www.zip-codes.com/zip-code-database.asp.

[2] http://wisconsin.hometownlocator.com.

[3] Eric S. Larsen, "Creating a Stored Macro Facility in Ten Minutes", *Proceedings of the SAS Global Forum 2008*, Paper 101-2008, at P. 1.

The FILENAME statement references the remote SAS® program, macro_zipcode.sas, which contains the macro %zipcode. This macro creates the variable COUNTY, based on the value of PAT_ZIP, or the patient zip code provided by the OASIS database. The %INCLUDE code invokes the %zipcode macro contained in macro_zipcode.sas. In effect it tells the SAS® to be prepared to merge the macro language at a later specified point into the calling program. These two lines can be combined into one line like this:

```
%INCLUDE "c:\stored_macros\macro_zipcode.sas";[4]
```

There are two other noteworthy bits of code in this program. One of these is mandatory for a SAS® program if it is to use %INCLUDE to call a remote macro and incorporate it. The other is optional but very nice to use for macro editing. The mandatory code segment is contained in the data step into which the macro is to be dumped:

```
DATA OASIS2;
SET TONY.OASIS_CALL;
%zipcode;
```

The optional coding is contained, appropriately enough, in this OPTIONS statement:

```
Options source2 /*nosource2*/ ORIENTATION=PORTRAIT;
```

SOURCE2 tells SAS® to include the macro statements in the log. This is very useful for debugging macros. Of course, with a macro of more than nine hundred lines, you probably do not want to keep dumping the macro language into the log once the code language is debugged. At that point you simply replace SOURCE2 with the commented out code, NOSOURCE2. The log will then merely enter this brief acknowledgment:

```
%macro zipcode;
```

As Appendix 1B shows, macro_zipcode.sas, the file containing the macro called by %INCLUDE, is quite spare. It is just a SAS® program file that serves as a shell for the actual macro language. There is no "OPTIONS" statement or external filename reference of any kind. This is because, when used with a %INCLUDE statement, the called SAS® file is merely a repository for the macro to be invoked. The calling SAS® program does all the work, compiling and executing the macro language contained in the called SAS® file. This simplicity is very attractive. However, it also means that every time this language is invoked with a %(macro name) command, it is recompiled and re-executed. Therefore, every time the program invokes the macro created by the %INCLUDE language to access the remote SAS® file containing the macro, the SAS program editor must use CPU time to check the macro for syntax, create temporary macro variables, and compile the macro. Depending on the size of the file being processed as a macro, this can pose a significant burden on overhead processing time.[5] For this reason, SAS® provides a second means of calling remote macros, the stored macro file facility.

---

[4] I owe discovery of this method of invoking %INCLUDE to Ronald Fehld of the Center for Disease Control and Prevention. See Ronald Fehl, "A SASAUTOS Companion: Reusing Macros", *Proceedings of SUGI 30,*Paper 267-30 at P. 5.

[5] See Erik S. Larsen, *Op. Cit.* at P. 1 for an excellent explanation of this problem, which I have largely incorporated in this paper.

OPTION 2: THE STORED, COMPILED MACRO FACILITY

I will start with the coding additions needed by both the calling SAS® program and the called SAS® file containing the macro in order to set up a stored macro facility. The calling program, contained in Appendix 2A, no longer has the %INCLUDE invocation or the FILENAME statement that identified the location of the macro_zipcode.sas, the file containing the zip code-to-county conversion macro. In their place we see a new OPTIONS statement and a new LIBNAME statement.

```
LIBNAME MACSTORE " C:\stored_macros";
OPTIONS MSTORE SASMSTORE=MACSTORE ORIENTATION=PORTRAIT;
```

The MSTORE and SASMSTORE options tell SAS® to create a stored macro library, MACSTORE, at the URL referenced in the LIBNAME statement. The actual command to merge the stored macro during the appropriate data step remains unchanged, even though the %zipcode macro is nowhere directly referenced in the call program:

```
DATA OASIS2;
SET TONY.OASIS_CALL;
%zipcode;
```

As noted in Appendix 2B, the SAS file containing the macro also has new language. This includes a LIBNAME statement that is identical to the one in the call program in Appendix 2A, and an OPTIONS statement that includes the same MSTORE and SASMSTORE options as are in the call program. In addition, the beginning of the %zipcode macro statement has new coding:

```
 %macro zipcode / store ;
If PAT_ZIP=53001 then county='SHEBOYGAN    ';
....
%mend zipcode;
```

The STORE keyword orders the SAS® system to store the macro source code in the MACSTORE library. These coding changes are the only additions and deletions that have to be made to change a %INCLUDE macro call routine into the programming for a stored, compiled macro. However, unless the SAS® file with the macro program, macro_zipcode.sas, is first submitted, running the call program in Appendix 2A will only give you this error message:

```
DATA OASIS2;
SET fall2008.patient;
....
%zipcode;
-
180
NOTE: The SAS System was unable to open the macro library referenced by the SASMSTORE = libref MACSTORE.
WARNING: Apparent invocation of macro ZIPCODE not resolved.
ERROR 180 – 322: Statement is not valid or is used out of proper order.
```

Submitting macro_zipcode.sas before running the call program the first time will create a file, SASMACR.SAS7BCAT. This file creates a subdirectory in the STOREMAC library and places the %zipcode macro there as a stored, compiled macro. The calling program will then be able to access STOREMAC, retrieve %zipcode and run it much more quickly and efficiently than would be the case using %INCLUDE. This is because it has already been stored as a compiled macro. The calling program file will even be able to access the %zipcode macro in subsequent SAS® sessions without first submitting macro_zipcode.sas. All that is necessary is that SASMACR.SASBCAT and macro_zipcode.sas remain in the same directory location as they were when macro_zipcode,sas was initially submitted, and that the LIBNAME statement creating STOREMAC and the MSTORE and SASMSTORE options in the calling program be in place and unchanged.

Just how much more quickly and efficiently the %zipcode macro will run as a stored, compiled macro can be seen by comparing the runtime for %zipcode using these two different macro calling tools. This is the log readout for how long it took for data step OASIS2 to run when it had to compile and execute a %INCLUDE command:

```
DATA OASIS2;
SET fall2008.patient;
....
%zipcode;
NOTE: There were 77149 observations read from the data set TONY.OASIS_CALL.
NOTE: The data set WORK.OASIS2 has 77149 observations and 37 variables.
NOTE: DATA statement used (Total process time):
      real time          3.17 seconds
      cpu time           0.51 seconds
```

And this is what the log shows for the same data step where %zipcode is run as a stored, compiled macro:

```
DATA OASIS2;
SET fall2008.patient;
....
%zipcode;
NOTE: There were 77149 observations read from the data set TONY.OASIS_CALL.
NOTE: The data set WORK.OASIS2 has 77149 observations and 37 variables.
NOTE: DATA statement used (Total process time):
      real time          0.56 seconds
      cpu time           0.53 seconds
```

While this data step processes a comparatively small number of records, we can see clear savings from the log. However, there is a tradeoff for this efficiency. Should you decide it was time to update the macro because, for instance, the Postal Service had reassigned some codes, simply typing your changes into macro_zipcode.sas is not going to change the code that the call program accesses. You must resubmit macro_zipcode.sas, thereby overwriting sasmacr.sas7bat. Even if you remember to do this,

stored macros are rather clunky to clean up, at least if you are counting on using the calling program's log for assistance. Therefore, when rewriting a remotely accessed macro, you may want to rewrite the call program so that it invokes your macro with a %INCLUDE call. You will also want to include SOURCE2 in the OPTIONS statement of your call program, of course. However, you won't need to go into macro_zipcode.sas and get rid of the STORE, MSTORE and SASMSTORE key words, or the LIBNAME statement creating the MACSTORE macro library. %INCLUDE does not use these options, and when it finds them in the called SAS® file it ignores them like so many empty calories.

SO WHICH IS IT: %INCLUDE OR THE STORED MACRO FACILITY?

This depends on your work environment and such things as the volume of data your office accesses at any given time. If you only plan on creating one or a few remotely accessed macros and overhead processing time is not an issue, you may prefer to stick with the simplicity of the %INCLUDE statement. This may particularly be the case if the remote macro in question requires frequent updating. However, if you expect to create several remotely accessed macros, you will probably create a common directory for these macro programs. Therefore, you might was well make it a library of stored, compiled macros. Addition considerations in some work environments include client or patient confidentiality, or the security of trade secrets. Confidential information may form part of the macro coding, and the employer may not want everyone who accesses the macro to be able to use SOURCE2 to read the macro language in the SAS® log. In that case, you're best invoking your remote macro with the stored macro facility option.

REFERENCES

Fehd, Ronald. 2005. "A SASAUTOS Companion: Reusing Macros", *Proceedings of SUGI 30,* Paper 267-30.

Huang, Jie and Lin, Tracy. 2009. "SAS® Macro Autocall and %Include", *Proceeding of SESUG 2009*, Paper CC-019.

Larsen, Erik S. 2008. "Creating a Stored Macro Facility in Ten Minutes", *Proceedings of the SAS Global Forum2008*, Paper 101-2008.

Steppe, Doris. 1998. "Introduction to the SAS® Macro Language," *Proceedings of NESUG 1998*.

Stojanovic, Mirjana, Watson, Dorothy and Hollis, Donna. 2006. "Utilizing the Stored Compiled Macro Facility in a Multi-user Clinical Trial Setting", *Proceedings of PharmaSUG 2006*, Paper AD05.

CONTACT INFORMATION

I value your comments and questions. Please feel free to contact me at

Name: Anthony Reeves
Enterprise: State of Wisconsin Department of Health Services
Address: 1 West Wilson Street, Room 950
City, State ZIP: Madison, WI 53707
Work Phone: (608) 267-3325
Work E-mail: anthony.reeves@wisconsin.gov

APPENDIX 1A: Call Program using %INCLUDE

```
/* This file uses a libname and the %INCLUDE option to access, compile and execute a macro */
/* No stored macro facility is created.*/

libname fall2008 'C:\OASIS_fall_08\data';

options source2 orientation=portrait nocenter pageno=1;
/*The source2 option incorporates the zip code macro “secondary source" read-out in the SAS*/
/*Log. If you don't want to do this, then use the nosource2 option */

filename zipcode "H:\SASv8\stored_macros\macro_inc_zipcode.sas";
%include zipcode;
/*The “zipcode” fileref statement and %include code identify the macro to be called. */

proc format;
  value $agegrp
  '01'='18 TO 29'
  '02'='30 TO 44'
  '03'='45 TO 54'
  '04'='55 TO 54'
  '05'='65 TO 74'
  '07'='85 TO 94'
  '06'='75 TO 84'
  '08'='95 AND OLDER'
;
  value $accident
  '0'='NO '
  '1'='YES';

data OASIS2;
 set fall2008.patient;
 if age>=18 and age<30 then age_group='01';
 if age>=30 and age<45 then age_group='02';
 if age>=45 and age<55 then age_group='03';
 if age>=55 and age<65 then age_group='04';
 if age>=65 and age<75 then age_group='05';
 if age>=75 and age<85 then age_group='06';
 if age>=85 and age<95 then age_group='07';
 if age>=95 then age_group='08';
 %zipcode; /%zipcode segment compiles the macro, and tells the program where to put the macro, which*/
 /*creates the new variable COUNTY.*/

proc sort; by county age fall;
```

APPENDIX 1A: Call Program using %INCLUDE (continued)

```
ods listing close;
ods pdf body='H:\SASv8\OASIS_fall_08\data\oasis_fall_2008_inc.pdf';
ods proclabel "HOME HEALH AGENCY PATIENTS IN 2008: FALLS RECORDED, YES OR NO?"; /*This gets rid of */
/*perfectly unnecessary information in the ODS PDF bookmark that identifies the source of this table as a */
/*Tabulate Procedure, and replaces it with useful information about the table and what it says.*/
title1 "WISCONSIN HOME HEALTH AGENCY PATIENTS: ";
title2 "BY COUNTY, AGE GROUP, AND WHETHER RESIDENT EITHER RECEIVED EMERGENCY";
title3 "CARE DUE TO AN ACCIDENTAL FALL SINCE THE PRECEDING OASIS ASSESSMENT,";
title4 "AS NOTED IN OASIS SECTION M0840, OR WERE BEING DISCHARGED FROM HOME";
title5 "HEALTH CARE TO A HOSPITAL DUE TO AN ACCIDENTAL FALL, AS NOTED IN";
title6 "OASIS SECTION M0890.";
footnote1 "PATIENTS RECEIVING AN OASIS ASSESSMENT MUST BE AT LEAST 18 YEARS OLD,";
footnote2 "MUST BE RECEIVING HOME CARE FROM A SKILLED HEALTH CARE PROVIDER, AND";
footnote3 "THEIR HOME HEALTH AGENCY MUST BE RECEIVING REIMBURSEMENT FOR SERVICES";
footnote4 "UNDER TITLE XVIII OR TITLE XIX.";

proc tabulate data=OASIS2 order=formatted format=comma8.  MISSING contents="";/*The contents="" code*/
/*segment gets rid of the "Cross-tabular summary report" statement from the ODS PDF bookmark. I view that*/
/*statement as useless clutter.*/
class county age_group fall;
tables county="COUNTY OF RESIDENCE"*(age_group="" ALL) all="STATEWIDE TOTAL", (fall all)*(n pctn<fall
all>*f=10.2) / MISSTEXT='NONE' RTS=25
BOX="OASIS 'RECORDS' ARE RANDOMLY GENERATED FROM CY 2008 ASSESSMENT FIELDS OF DIFFERENT PATIENTS,
AND MAY ONLY BE USED FOR SYSTEM TESTING" INDENT=5 contents=""; /*The contents="" code segment gets */
/*rid of the "Table 1" statement, which I also find to be useless clutter to the ODS PDF bookmark statement.*/
  KEYLABEL N='COUNT'
        PCTN='PERCENT'
                        ALL='TOTAL';
  LABEL    FALL='IS FALL NOTED AT OASIS M0840/M0890?';
FORMAT     FALL $accident.
        age_group $agegrp.; run;

ODS PDF close;
ODS LISTING;
```

APPENDIX 1B: SAS Program Called Using %INCLUDE

```
%macro zipcode;
If pat_zip=53001 then county='SHEBOYGAN    ';
If pat_zip=53002 then county='WASHINGTON ';
If pat_zip=53003 then county='DODGE          ';
If pat_zip=53004 then county='OZAUKEE      ';
If pat_zip=53005 then county='WAUKESHA    ';
If pat_zip=53006 then county='DODGE          ';
. . . . . .
If pat_zip=53982 then county='WAUSHARA    ';
If pat_zip=53983 then county='WAUPACA      ';
If pat_zip=53984 then county='WAUSHARA    ';
If pat_zip=53985 then county='WINNEBAGO  ';
If pat_zip=53986 then county='WINNEBAGO  ';
If pat_zip=53987 then county='WAUPACA     ';
%mend;
```

APPENDIX 2A: Call Program Using a Stored Macro Facility

```
libname fall2008 'C:\OASIS_fall_08\data';

LIBNAME MACSTORE 'C:/STORED_MACROS'; /*this LIBREF statement identifies the location of the stored macro*/
/*facility*/

OPTIONS MSTORE SASMSTORE=MACSTORE  ORIENTATION=PORTRAIT;  /* MSTORE tells the SAS® System to look*/
/* for sasmacr.sas7bcat, which catalogs, compiles and stores the macros in the macro facility.  SASMSTORE */
/*references the LIBREF that has the macros compiled and stored by sasmacr.sas7bat. */

PROC FORMAT;
  VALUE $agegrp
 '01'='18 to 30'
 '02'='31 to 45'
 '03'='46 to 55'
 '04'='56 to 65'
 '05'='66 to 75'
 '06'='76 to 85'
 '07'='86 to 95'
 '08'='96 AND OLDER';
 VALUE $accident
 '0'='NO '
 '1'='YES';

DATA OASIS2;
SET fall2008.patient;
if age>=18 and age<30 then age_group='01';
 if age>=30 and age<45 then age_group='02';
 if age>=45 and age<55 then age_group='03';
 if age>=55 and age<65 then age_group='04';
 if age>=65 and age<75 then age_group='05';
 if age>=75 and age<85 then age_group='06';
 if age>=85 and age<95 then age_group='07';
 if age>=95 then age_group='08';
%zipcode; /*the macro language is inserted and compiled here: it creates a new variable, COUNTY, with a value*/
/*that depends on the home health care patient's residential zip code. */

PROC SORT DATA=OASIS2; BY county;

ODS LISTING CLOSE;
ODS PDF BODY=OUT;
ODS proclabel "HHA PATIENTS W. FALLS IN THE HOME"; /*I find a bookmark that explains what your table is */
/*about  to be more useful marginal information that one that tells you a Frequency Procedure was used to */
/*create it*/
```

APPENDIX 2A: Call Program Using a Stored Macro Facility (continued)

```
title1 "WISCONSIN HOME HEALTH AGENCY PATIENTS: ";
title2 "BY COUNTY, AGE GROUP, AND WHETHER RESIDENT EITHER RECEIVED EMERGENCY";
title3 "CARE DUE TO AN ACCIDENTAL FALL SINCE THE PRECEDING OASIS ASSESSMENT,";
title4 "AS NOTED IN OASIS SECTION M0840, OR WERE BEING DISCHARGED FROM HOME";
title5 "HEALTH CARE TO A HOSPITAL DUE TO AN ACCIDENTAL FALL, AS NOTED IN";
title6 "OASIS SECTION M0890.";
footnote1 "PATIENTS RECEIVING AN OASIS ASSESSMENT MUST BE AT LEAST 18 YEARS OLD,";
footnote2 "MUST BE RECEIVING HOME CARE FROM A SKILLED HEALTH CARE PROVIDER, AND";
footnote3 "THEIR HOME HEALTH AGENCY MUST BE RECEIVING REIMBURSEMENT FOR SERVICES";
footnote4 "UNDER TITLE XVIII OR TITLE XIX.";
proc tabulate data=OASIS2 order=formatted format=comma8.  MISSING contents="";/*The contents="" code*/
/*segment gets rid of the "Cross-tabular summary report" statement from the ODS PDF bookmark. I view that*/
/*statement as useless clutter.*/
class county age_group fall;
tables county="COUNTY OF RESIDENCE"*(age_group="" ALL) all="STATEWIDE TOTAL", (fall all)*(n pctn<fall
all>*f=10.2) / MISSTEXT='NONE' RTS=25 BOX="OASIS 'RECORDS' ARE RANDOMLY GENERATED FROM CY 2008
ASSESSMENT FIELDS OF DIFFERENT PATIENTS, AND MAY ONLY BE USED FOR SYSTEM TESTING" INDENT=5
contents=""; /*The contents="" code segment gets rid of the "Table 1" statement, which I also find to be */
/*useless clutter to the ODS PDF bookmark statement.*/
  KEYLABEL N='COUNT'
        PCTN='PERCENT'
                        ALL='TOTAL';
  LABEL    FALL='IS FALL NOTED AT OASIS M0840/M0890?';
FORMAT     FALL $accident.
        age_group $agegrp.; run;

ODS PDF close;
ODS LISTING;
```

APPENDIX 2B: SAS Program Called Using a Stored Macro Facility

```
/*Remember to submit this program BEFORE you submit the call program in Appendix 2A the first time*/

LIBNAME macstore 'c:\stored_macros'; /*/THE LIBNAME statement and the MSTORE and SASMSTORE  */
OPTIONS MSTORE SASMSTORE=macstore; /*options reference the library containing this macro and order the */
/*SAS® System to place sasmacr.sas7bcat at this file location.*/

%MACRO zipcode / STORE ; /*The STORE macro option orders that the catalogue sasmacr.sas7bcat be created.  */
If pat_zip=53001 then county='SHEBOYGAN    ';
If pat_zip=53002 then county='WASHINGTON ';
If pat_zip=53003 then county='DODGE          ';
If pat_zip=53004 then county='OZAUKEE      ';
If pat_zip=53005 then county='WAUKESHA    ';
If pat_zip=53006 then county='DODGE          ';
. . . . . .
If pat_zip=53982 then county='WAUSHARA    ';
If pat_zip=53983 then county='WAUPACA      ';
If pat_zip=53984 then county='WAUSHARA    ';
If pat_zip=53985 then county='WINNEBAGO  ';
If pat_zip=53986 then county='WINNEBAGO  ';
If pat_zip=53987 then county='WAUPACA     ';
%mend;
```