

Short, Sweet and Simple...how to do more with less in SAS®.

Brian Spruell, SciMetrika, LLC, RTP, NC

ABSTRACT

This paper will demonstrate a technique the author learned using %str and %scan which greatly reduces the amount of actual programming needed to perform a task in SAS®. He will do this by presenting several examples from his work as a government contractor. This paper is intended for beginner/intermediate SAS programmers.

INTRODUCTION

While I often look for ways to automate the programs I am working on macro variables have become invaluable resources to me over the past seven years as a SAS programmer. I have found that automation greatly reduces the risk of human error both in terms of program maintenance and execution.

I hope to demonstrate the usefulness of the %str and %scan macro functions by presenting how I used them in a recent collaboration with researchers at the CDC. This technique proved to be useful considering the nature of the project which required a lot of collaboration going back and forth between the researchers and myself. It helped to reduce the amount of changes I needed to make in order to fulfill the researchers' requests. It kept the number of changes within a few particular places within in the program which increased my turn around time.

The project that required me to use these functions was a contextual breast cancer study with the CDC. The CDC wanted to look at the relationship between certain factors (age, poverty level, number of teaching hospitals in area, etc.) and the quality of treatment a patient received for breast cancer (surgery, radiation therapy, etc.). They wanted me to adjust the model for all these terms and produce odds ratios.

MACRO FUNCTIONS

SAS comes with several built in macro functions. You can use these macro functions in open code and also in macro definitions. The table below has been pulled from the SAS 9.1.3 documentation which lists all the macro functions and their descriptions. I have bolded the two macro functions we will be discussing in this paper:

Function	Description
%BQUOTE, %NRBQUOTE	mask special characters and mnemonic operators in a resolved value at macro execution.
%EVAL	evaluates arithmetic and logical expressions using integer arithmetic.
%INDEX	returns the position of the first character of a string.
%LENGTH	returns the length of a string.
%QUOTE, %NRQUOTE	mask special characters and mnemonic operators in a resolved value at macro execution. Unmatched quotation marks (" ") and parentheses (()) must be marked with a preceding %.
%SCAN, %QSCAN	search for a word specified by its number. %QSCAN masks special characters and mnemonic operators in its result.

%STR, %NRSTR	mask special characters and mnemonic operators in constant text at macro compilation. Unmatched quotation marks (" ") and parentheses (()) must be marked with a preceding %.
%SUBSTR, %QSUBSTR	produce a substring of a character string. %QSUBSTR masks special characters and mnemonic operators in its result.
%SUPERQ	masks all special characters and mnemonic operators at macro execution but prevents resolution of the value.
%SYMEXIST	returns an indication as to whether the named macro variable exists.
%SYMGLOBL	returns an indication as to whether the named macro variable is global in scope.
%SYMLOCAL	returns an indication as to whether the named macro variable is local in scope.
%SYSEVALF	evaluates arithmetic and logical expressions using floating point arithmetic.
%SYSFUNC, %QSYSFUNC	execute SAS functions or user-written functions. %QSYSFUNC masks special characters and mnemonic operators in its result.
%SYSGET	returns the value of a specified host environment variable.
%SYSPROD	reports whether a SAS software product is licensed at the site.
%UNQUOTE	unmasks all special characters and mnemonic operators for a value.
%UPCASE, %QUPCASE	convert characters to uppercase. %QUPCASE masks special characters and mnemonic operators in its result.

%STR

The %str macro function is extremely useful for those character strings which contain the following:

- 1) A semicolon will be treated as text instead of being treated as part of the macro statement
- 2) The blanks after the %str function are used
- 3) Quotation marks

In addition to the adjusted odds ratios I was asked to produce unadjusted odds ratios. This would require fitting the outcome to each variable individually. Originally I was looking at 22 terms which required coding 22 separate logistic regressions. Though my use of %str and %scan I was able to just write one logistic model and fit each of those terms separately. The example below is taken from our final model program when the variables were reduced to 17.

The first step for my logistic macro was to create a macro variable called vars which listed all the 17 variables I was interested in using in my model. I called the macro analysis.

```

%macro analysis;
  %let vars=%str( age_c
                  _RaceRec
                  hisprec_r
                  marstat2
                  _Rural_Urban_Code
                  Percent_Below_Poverty_Level2
                  _pctelig
                  dich_hisp_female
                  dich_blk_female
                  _Rad_Onc_2004
                  _Gen_Sur_2004
                  ST_Hosp_Medical_School_2004r
                  derstage2_
                  tumor_size
                  receptor
                  dxyear
                  regid_new
  );

```

I've found it easier to list the variables in the manner I have listed above which makes it easier to add and remove variables as needed. I could list them in any order I desired. I just have to keep a space between each variable since that is what the %scan function will look for when parsing out these variables.

%SCAN

The next step in our macro code is to use the %scan function in order to parse out each of the variables listed in the vars macro. I used a macro do loop in order to go through each variable and add them to the model:

```

%do i=1 %to 17;
  %let var=%scan(&vars.,&i,' ');

  proc logistic data=analysisds;
    ods output Oddsratios=Oddsratios;
    ods output ParameterEstimates=ParmEST;
    model surgery (event="1") = &var./link=logit;
  run;
%end;

```

Each iteration of the doo loop will assign one of the variables listed in vars into the new macro variable var which is used in the logistic code. The output is then saved in one of the ods output tables (Oddsratios which contains the odds ratios and the Parmest dataset which contains p-values produced by the logistic procedure).

The first element of the %scan function is the list of variables contained in the macro variable vars. The next element is the number from the do loop which tells scan where in vars to pull the next variable. The final element tells the %scan function how to differentiate each member in the list. For instance if each element in vars was separated by a ',' instead of a ' ' then we could have written the %scan function as follows:

```

%let var=%scan(&vars.,&i,',');

```

This can be especially helpful when the list contains phrases instead of just words. In other projects I have wanted to produce a list of labels to apply to newly created variables. Since the labels themselves contain spaces I would have to use commas to separate the labels instead of spaces as below:

```

%let labels=%str( Mean Age in Years, Mean Age in Months, ID number) ;
%let label=%scan(&labels.,&i,',');

```

The above code would go through three iterations. After each iteration the appropriate label would be applied to the variables.

For instance the code: label avgyr = "&label"; for the first iteration would resolve to

Label avgyr = 'Mean Age in Years';

DIFFERENT OUTCOMES

At one point in this project the CDC wanted us to look at a variety of potential outcomes which required me to add another do loop around the already existing do loop.

First I created a new macro variable with the %str function which contained all the outcomes the CDC was interested in:

```
%let outcomes=%str(surgery radtrt_bcs primary_trt);
```

The values of the outcome variables differed from variable to variable so I had to add a new macro variable that contained the list of events the CDC wanted me to model:

```
%let events=%str(2 1 1);
```

I then added an outer do loop and used the %scan function to assign each outcome to the model:

```
%do j = 1 %to 3;
    %let outcome=%scan(&outcomes, &j, ' ');
    %let event=%scan(&events, &j., ' ');

...
proc logistic data=analysisds;
    ods output Oddsratios=Oddsratios;
    ods output ParameterEstimates=ParmEST;
    class &var. (ref="&reference.") /param=ref;
    model &outcome. (event="&event.") = &var./link=logit;
run;
```

The reference macro is the same as event and outcome macros. It represents each reference level for all the terms added to the logistic macro.

OUTPUT DATASET

Each iteration the logistic procedure produces the odds ratios into the oddsratios dataset from the ods output statement. Since the data is overwritten with each iteration I placed the output into a new dataset which is subset with each iteration:

```
data out;
    %if (&i=1) %then set; %else set out.; OddsRatios;
run;
```

With the first iteration (&i=1) I create the dataset out by setting just the OddsRatios dataset. For all subsequent iterations, I set both out and the new OddsRatios dataset.

The out data is then manipulated so that rows are given appropriate headers and the odds ratios are set to the appropriate length. I created a format which would be the row label for each of the 17 variables:

```
Proc format;
```

```
value rowheadfmt
    1 = 'Age(years)^(super 3)'
    2 = 'Race^(super 3)'
    3 = 'Hispanic Ethnicity^(super 3)'
    4 = 'Marital Status^(super 6)'
    4.1 = 'Area Based Characteristics'
    5 = 'Rural/Urban for county^(super 4)'
    6 = 'Percent living below poverty level^(super 1)'
```

```

7 = 'Percent of Uninsured Females (county level)5'
8 = 'Percentage of Hispanic females1'
9 = 'Percentage of Non-Hispanic black females1'
9.1 = 'Medical Resources at County Level'
10 = 'Radiation Oncologist per 100,000 females1'
11 = 'General Surgeons per 100,000 females1'
12 = 'Medical School (2004)1'
12.1 = 'Breast Cancer Characteristics'
13 = 'Stage (AJCC)3'
14 = 'Tumor Size (cm)3'
15 = 'Estrogen and Progesterone receptor3'
15.1 = 'Registry Data'
16 = 'Dx Year3'
17 = 'SEER Registries3'
;

```

Those formats were then feed into the row variable in the out dataset

```

data out;
    length row $55;
set out;
    by n;
    row=put(&i., &rowheadfmt.);
run;

```

Finally the out dataset was sent to an rtf file using ods rtf and proc report:

```

ods rtf file="&basedir.\Quality of Treatment\Output\Tables For
Paper\final\2.1.OddsRatios_Final_%sysfunc(date(),mmddyy6.).rtf" STYLE=custom
/*notoc*/;

```

```

proc report data=outputds split='*' headline headskip spacing=2 nofs nowd;
    column ("Independent Variable" row_label) cell_n_3 cell_n_sub3 cell_per3
("Adjusted" cell_adjor_3 cell_adjCI_3);
    COMPUTE row_label;
        IF index(row_label, "^")=0
        THEN CALL DEFINE(_COL_, "style", "STYLE=[font_weight=bold]");
    ENDCOMP;
    compute after;
        line ' Definitive Local Therapy defined as receiving
Mastectomy or Breast Conserving Surgery with radiation Therapy';
        line 'Adjusted Odds Ratios are adjusted for age, race,
hispanic ethnicity, marital status, education, ';
        line ' Rural/Urban, Percent living below poverty, percent
uninsured female, percentage of hispanic females, ';
        line ' Percentage of non-Hispanic black females, radiation
oncologist per 100,000 females, general surgeons';
        line ' per 100,000 females, Medical School, NCI-Designated
Comprehensive Cancer Center, Stage, Tumor Size, ';
        line ' estrogen and progesterone receptor, dx year, and
SEER registries';
        line '1: Area Resource File';
        line '2: NCI Website';
        line '3: SEER';
        line '4: US Dept. of Agriculture Economic Research
Service';
        line ' Metropolitan includes codes 01, 02, 03. Urban
includes codes 04, 05, 06, 07. Rural includes codes 08, 09.';

```

```

line '5: 2005 Health Insurance Coverage Status for States
and Counties from Census';
line '6: Marital Status Other: Includes separated,
divorced, and widowed';
endcomp;
/* title 'Table 4.1 Odds Ratios for (Mastectomy or Partial with
Radiation therapy) vs (Just Parital)';*/
/* title 'Table 2.1 Odds Ratios for Definite Therapy(either
Mastectomy or Breast Conserving Surgery with radiation) vs (Breast Conserving
Surgery without radiation)';*/
title 'Final Table 2.1 Odds Ratios for Receiving Definitive
Local Therapy';
footnote 'Page ^{pageof}';
run;
ods rtf close;

```

CONCLUSION

I have been able to use this technique for many different projects. These techniques can be adapted for more than just regression analysis. For example, in the same project I used it for producing the characteristics table for the paper which looked at cross frequencies between each term and the outcome variables. Using these two macro functions has greatly reduced the amount of time I write the actual program.

REFERENCES

"SAS® 9.1 Macro Language: Reference". SAS OnlineDoc 9.1.3. 2006. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Brian Spruell
 SciMetrika, LLC
 100 Capitola Drive, Suite 104
 Research Triangle Park, NC 27713
 (919) 544 - 8988
 bspruell@scimetrika.com

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.