Macros for Calculating Block and Rolling Averages

Randy C. Finch, Tennessee Valley Authority, Muscle Shoals, AL 35662-1010

Abstract

This paper describes two macros that work together to generate either block or rolling averages of user-specified variables in user-specified data sets. The code needed before executing the macros is also shown and explained.

Introduction

My SAS® applications generate daily data sets containing a large amount of plant process data. Typically, there are about 200 variables in each data set along with a date-time variable. One observation is generated every 15 minutes. Some time ago, I was asked to create a SAS/AF® application that combined user-selected data sets, sorted the combined data by date-time, and then subjected user-selected variables to a block or rolling average analysis. Block averages are non-overlapping averages, whereas rolling averages are overlapping. For example, block averages might contain six-hour averages for times 12:00AM-5:45AM, 6:00AM-11:45AM, 12:00PM-5:45PM, etc. Rolling averages might contain six-hour averages for times 12:00AM-5:45AM, 1:00AM-6:45AM, 2:00AM-7:45AM, etc. I decided to use the macro facility for accomplishing this task.

The Macros

Two macros, DOAVG and LOOPAVG, were created for this task. Both are shown in Listing 1. The DOAVG macro is responsible for extracting the data for an appropriate time span, generating the average for each variable, and combining the results with previously generated averages. The LOOPAVG macro is responsible for "walking through" the data using a loop and calling the DOAVG macro. LOOPAVG is the macro that is executed in the program proper.

The macros assume that the date-time variable is named DTIME. Also, they make use of several preassigned macro variables. They are:

BEGTIME	Initially assigned the value of the first DTIME value in the combined sorted data set. This value changes each time the DOAVG macro executes such that it always reflects the first DTIME value for the next average calculation.
ENDTIME	The value of the last DTIME value in the combined sorted data set. This value never changes.
STEPTIME	The number of minutes between averages. It must be a multiple of 15 since the data is taken on the quarter hour. For a block average, this is the amount of time between the last DTIME of one average and the first DTIME of the next average and will normally be zero. For a rolling average, this is the amount of time between the first DTIME of one average and the first DTIME of the next average. It must be greater than zero or the macros will get caught in an infinite loop.
AVGTIME	The amount of time, in minutes, over which each average will be taken.
MULTFACT	This equals zero if a rolling average is needed and one if a block average is needed.
VARNAMES	A string of variable names separated by spaces. Averages will only be taken for the variables in this list.
PRNUMVAR	The number of variables in the VARNAMES list.

The LOOPAVG macro executes a %DO %WHILE loop until there is not enough observations left in the data set to cover the AVGTIME requirement. It does this by comparing ENDTIME with the sum of the current BEGTIME (which is in units of seconds) and AVGTIME times 60 (to convert minutes to seconds) minus 900 (the number of seconds in 15 minutes). As long as ENDTIME is greater than or equal to the latter term, the loop will execute. The only activity in the loop is to call the DOAVG macro.

The DOAVG macro has three steps. The first is a data step that takes a previously created data set, WORK.COMBINED, which contains the sorted combined data sets, subsets it using an IF statement so that it only contains the data needed for the current average, and then stores the subset in another data set, WORK.TEMP. When the end of the WORK.COMBINED data set is reached, BEGTIME is updated to the first DTIME value for the next average by adding STEPTIME and AVGTIME(after converting each to seconds). When a rolling average is needed, MULTFACT will equal zero, preventing AVGTIME from being added to the old BEGTIME. This will result in overlapping averages.

The second step is the MEANS procedure. The procedure is only applied to the variables specified in the VARNAMES list. The means and numbers of observations used to calculate them are generated. The results are saved in the WORK.AVGTEMP data set.

The third and final step is the APPEND procedure. WORK.AVGTEMP is appended to WORK.AVGMAIN. When the first average is generated by PROC MEANS, WORK.AVGMAIN does not exist. SAS creates an empty data set and appends WORK.AVGTEMP. Each time this macro executes, a new observation is added to WORK.AVGMAIN. When the LOOPAVG macro finishes, WORK.AVGMAIN will contain all the averages of the designated variables for the entire WORK.COMBINED data set.

The Code That Uses the Macros

Several steps are involved in preparing for the execution of the LOOPAVG macro. First, the names of the data sets to be combined for averaging have to be assigned to the DATASETS macro variable, and the names of the variables to be used need to be assigned to the VARNAMES macro variable. The number of variables in the

VARNAMES list needs to be assigned to the PRNUMVAR macro variable. The MULTFACT macro variable needs to be assigned a value of zero if a rolling average is needed or a value of one if a block average is needed. The AVGTIME and STEPTIME macro variables also need to be assigned appropriate values. Listing 2 shows these macro variables being assigned using %LET statements. However, if the remaining code is part of a SUBMIT block within the SCL code of a SAS/AF application, the variables can represent SCL variables outside the SUBMIT block.

After assigning the above macro variables, a data step is used to combine all the designated data sets into the WORK.COMBINED data set, keeping only the designated variables and DTIME. The SORT procedure is then used to sort the combined data by DTIME.

Next, a _NULL_ data step is used to extract the first and last DTIME values in the combined data and assign these values to the macro variables BEGTIME and ENDTIME, respectively.

Next, the DATASETS procedure is used to delete the WORK.AVGMAIN data set that will be used by the DOAVG macro for storing the averaged data. This prevents the new set of averages from being combined with a previous set of averages.

Finally, the LOOPAVG macro is executed. After it completes, the WORK.AVGMAIN data set will contain all the averaged data.

Conclusion

These macros have been very effective at generating block and rolling averages of data. However, they are rather slow on older computers when generating averages for combined data sets having hundreds of observations. Some of the code in these macros is specific to my application such as assuming the date-time variable is named DTIME and that the data will be in 15 minute increments. This can be easily changed for other applications or generalized by adding more macro variables.

Listing 1 - Macros for Creating a Data Set With Block or Rolling Averages

```
%MACRO DOAVG;

  DATA WORK.TEMP;
    SET WORK.COMBINED END=EOF;
    DROP NEWTIME;
    IF EOF THEN DO;
       NEWTIME = &BEGTIME + &STEPTIME*60 +
                 &MULTFACT*&AVGTIME*60;
       CALL SYMPUT('BEGTIME',NEWTIME);
    END;
    IF (DTIME >= &BEGTIME) AND
       (DTIME <  &BEGTIME + &AVGTIME*60);
  RUN;

  PROC MEANS NOPRINT IDMIN MEAN N;
    VAR &VARNAMES;
    ID DTIME;
    OUTPUT OUT=WORK.AVGTEMP
           MEAN(&VARNAMES)=&VARNAMES
           N(&VARNAMES)=N1-N&PRNUMVAR;
  RUN;

  PROC APPEND BASE=WORK.AVGMAIN DATA=WORK.AVGTEMP;
  RUN;

%MEND;

%MACRO LOOPAVG;

   %DO %WHILE(&ENDTIME >= &BEGTIME + &AVGTIME*60
                          - 900);
      %DOAVG;
   %END;

%MEND;
```

Listing 2 - SAS Code Using the Macros

```
%LET DATASETS=JUN01 JUN02 JUN03;
%LET VARNAMES=AFG_IVEN AFG_OVEN;
%LET PRNUMVAR=2;
%LET MULTFACT=1;
%LET AVGTIME =180;
%LET STEPTIME=0;

DATA WORK.COMBINED;
   SET &DATASETS;
   KEEP DTIME &VARNAMES;
RUN;

PROC SORT;
   BY DTIME;
RUN;

DATA _NULL_;
  SET WORK.COMBINED END=EOF;
  IF _N_=1 THEN CALL SYMPUT('BEGTIME',DTIME);
  IF EOF THEN CALL SYMPUT('ENDTIME',DTIME);
RUN;

PROC DATASETS LIBRARY=WORK;
  DELETE AVGMAIN;
RUN;

%LOOPAVG;
```