

Detection and quantification of asymmetric peaks

Carry W. Croghan and Ronald W. Williams, US-EPA, Research Triangle Park, NC

ABSTRACT

One of the goals of a recent U.S. Environmental Protection Agency air pollution study was to relate the participants' activities to their exposures to particulate matter (PM) during their routine activities. PM concentration data were collected continuously with a real-time monitor, downloaded daily in CSV format and concatenated using AWK. Human activity diary information was collected via pen and paper and recovered using scannable decoding software (TeleForm). All data were converted into SAS. The asymmetric response peaks of the particulate matter concentrations made systematic detection and analysis of the area under their respective curves problematic. To overcome this problem, a program was written to locate each peak and generate an integrated value for its response area. Data were smoothed using a running average, inflection points were identified relative to the start and stoppage of each peak, and a moving baseline calculated. Incorporation of human activity descriptors allowed for the quantification of human activity pattern influence upon potential exposures to particulate matter under real-world conditions.

INTRODUCTION

Many daily activities affect the level of particulate matter (PM) in the air. These activities include cleaning (dusting, sweeping and vacuuming), cooking and personal hygiene (using spray products like hair spray, deodorant and perfume). Increased levels of PM have been associated with various health effects including cardio-pulmonary and respiratory diseases. We needed to determine the exposure level of this pollutant and associate that level with daily activities of a subpopulation of cardiovascular-impaired residents of central North Carolina. PM mass concentrations are measured on a semi-continuous scale. One means of estimating PM mass concentrations is determining the area under the curve using a continuous monitor. Graphing the data over time, it is easy to locate major PM contributions to an individual's personal exposure by noting the "peaks" generated by the monitor. Graphically, these can be easily seen. Locating them programmatically is a different matter. Most programs written to locate peaks assume a symmetric peak. The beginning and ending of each peak are indicated by a change in the sign of the slope of the tangent line to the curve. These points of change are commonly known as inflection points. Once the slope of the tangent line to the curve is zero, the top of the peak is assumed. The area under the front and rear portions of the halves under the curve are assumed to be equal. In PM mass concentrations, an exponential decline of the rear half of the peak is often observed, the results from some integration programs greatly underestimated the true area under the curve.

We will be discussing two elements in this paper. The first discussion point is the conversion of the data from multiple sources into SAS. The data were collected in various methods including paper questionnaires, computer assisted technician interview, and electronic transfer from the monitoring instrument. The second element is the program that was written to locate the peaks in the data.

DATA TRANSFERS

The first and often the most arduous step of any analysis is getting the data into the proper structure and format. Since various methods of data capture were used, each type of data will be examined separately.

PM DATA

Exposure monitoring data were electronically downloaded daily from the monitoring instrument in a comma separated value format (CSV). Thus one file per day per participant was generated. A sample of the data follows:

```
pDR-1000, , ,
User ID: 09841S20010308
Tag Number: 01, ,
Number of logged points: 1438
Start time and date: 08:11:00 08-Mar
Elapsed time: 23:58:00,
Logging period (sec): 60,
Calibration Factor (%): 100.0,
Max Display Concentration: 0.371 mg/m3
Time at maximum: 11:23:03 Mar 08
```

```

Max STEL Concentration: 0.155 mg/m3
Time at max STEL: 11:33:00 Mar 08
Overall Avg Conc: 0.025 mg/m3,
Logged Data: , ,
Point, Date , Time , Avg.(mg/m3)
1, 8-Mar, 08:12:00, 0.0001
2, 8-Mar, 08:13:00, 0.0021
3, 8-Mar, 08:14:00, 0.0021
4, 8-Mar, 08:15:00, 0.0011
5, 8-Mar, 08:16:00, 0.0021
6, 8-Mar, 08:17:00, 0.0021
. . .
1437, 9-Mar, 08:08:00, 0.0071

```

The example above illustrates the two levels of information stored in each file. The first type of information is summary statistics provided automatically by the monitoring instrument. The second type of information is the individual records of concentration levels observed during the sampling event.

In general, CSV files are easily read into SAS using a variety of techniques. One difficulty in processing these files is the non-uniform header that appears at the beginning of each file. The varying size of the header information caused the rows in which the data were stored to also vary. The header was the only place where the critical information, the participant identification number, was recorded. These problems could have been tackled within the SAS dataset using conditional input statements. However, AWK, a line parsing program, (Wikipedia) seemed to be a more straightforward method of dealing with the problem. It is very similar to PERL. It processes each line of a file and performs the commands. Although the length of the header varied, there were some important consistencies. The line that held the participant identification number always started with the word 'User'. The identification number was always the third word in that line. The data were always preceded by a line that started with the word 'Point'. There was never anything other than data after the data started being printed out. These elements were keyed on in the development of the AWK program.

AWK programs consist of a series of lines that have a pattern and an associated command. The patterns can be a set of conditions or a regular expression. The commands are combinations of function calls, variable assignments, and calculations. The only function that is used in our program is the print function.

```

# Awk program : mie
#   purpose : Strip off header and keep data
#

BEGIN                                {printit=0}
/User/                              {pid=$3}
$1 ~ /Point/                         {printit=1}
$1 !~ /Point/ && printit==1          {print pid ", " $0}
END                                  { }

```

Stepping through this AWK program, the lines that start with '#' are comments. It is good practice to start a program with a BEGIN statement. This is a special condition that says that a program has begun. Initial values of variables are set with the BEGIN statement. In this situation, the indicator variable, PRINTIT, was set to 0. If the word 'User' was in the line, then the third word was saved as the participant identification (PID). If the first word in the line was 'Point', then the indicator variable PRINTIT was set to 1. If the indicator variable was 1 and the first word was not 'Point', then the variable PID followed by a comma and the entire line was printed out. The END statement is useful for two things. First it clearly marks the end of the program, which is a good programming practice. The second and real intention is to issue any final commands once the end of the input file is reached.

The AWK program was run in a batch with all the files being read in and a single file being created.

```

awk -f "C:\programs\mie.awk" "D:00141S20010102.CSV" > "D:mie.txt"
awk -f "C:\programs\mie.awk" "D:00141S20010103.CSV" >> "D:mie.txt"
...
awk -f "C:\programs\mie.awk" "D:09841S20010311.CSV" >> "D:mie.txt"

```

The line commands used for running this AWK program consisted of 6 things. It must start with the program call to AWK, which is 'awk'. The '-f' tells AWK the program is stored in a file instead of being issued on the command line. The next element is the program file name, followed by the name of the input file. The single '>' instructs AWK to output to a file and replace any existing file. The double '>' instructs AWK to append the information to an existing file. The final element is the name of the output file itself.

Once the data were in a single file, the data were transferred into SAS using a simple input command.

ACTIVITY DIARY DATA

Activity diary information was collected from adult volunteer participants via pen and paper and recovered using scannable decoding software (TeleForm). Each batch run of the software generated five separate files, one for each page of the form. Since the files were of a uniform structure, converting the data into SAS was easy. The dataset was at the participant-day level. Each time-block had two variables: description of a human activity and location of the participant. The first step was to transpose the data into participant, date, time structure. There was only one date associated with each form. However, the data collected in a single form represented more than one day. Each day of data collection, the participant was given a new diary for the next 24 hour period. There were instances where the information for a specific date/time combination was filled out by the participants on two different forms. All this information needed to be consolidated into a single record per participant/date/time combination with no false replication of time points. The duplicate records were printed out and compared. If the information contained in the records was the same, then one record was kept. If the information was different, the descriptions were concatenated.

The difficulty in processing the activity data lie in the interpretation and coding of the data. For every 15 minutes of each 24 hour period, the participants recorded what they were doing on their diaries. That information had to be converted into dichotomous variables indicating whether certain activities were done. These activities of interest were COOKING, CLEANING and PERSONAL GROOMING. Many attempts were made to program the coding of these variables. However, this proved to be unsuccessful. Instead, a file was generated that included each unique activity description. There were over 13 thousand unique descriptions. These activities descriptions were read and coded by two individuals. The coding was compared and discrepancies were discussed until consensus was achieved as to the correct coding. There were 2547 activity descriptions that were classified as COOKING. This is a sample of some of the activity descriptors:

Activity Description

ARRIVE AT DINER
ASSEMBLE HAMBURGER HELPER
BACK OVER THE STOVE AGAIN
BACK TO BAKERY
BEGAN SUPPER
BETWEEN STEAMING
BISCUTS BAKING
BOIL COFFEE ON STOVE
BROIL
BROILING
BROWN HAMBURGER
BUTTERBEANS
COMPLETE DINNER
CROQUETTES
FINISH - OVEN OFF
FINISH COOKING
FIX SOMETHING TO EAT
FOOD SIMMERING
FRIED AND BOIL
FRIENDSHIP CAKE
FRY FISH
FRYING
HEAT LEFTOVERS

Activity Description

HEATED COFFEE
KITCHEN, COFFEE BREWING
LIT OUTDOOR GRILL
MADE PIES
PREPARE BREAKFAST
PREPARING A SNACK
PUT CAKE IN OVEN
REHEAT FISH
RESTAURANT
SEARED MEAT, COOKING
SEASON CHICKEN
SIMMERING
SQUASH
STARTED DINNER
STEAM CABBAGE
STEWING
STIRRING FOOD ON STOVE
STOVE
TOAST
TOAST BREAD
TOASTER OVEN
WARM BURGERS

Often the entire phrase would have to be read in order to determine if cooking had occurred. Many times the participant stated what they were not doing as well as what they were doing. For example, one participant would write "IN KITCHEN, NO COOKING". In addition, several verbs like fixing, making, and starting were not unique to just one activity. Therefore, it was only after reading the entire phrase that we could determine whether cooking had actually occurred. It was decided that any mention of a restaurant or a food item would result in a cooking code. Deciding the inclusion words/phases for each activity made it easier to do the coding. Of course, there were some interpretation differences. This was why it was critical to have the coding done by more than one person.

FOLLOW-UP QUESTIONNAIRE

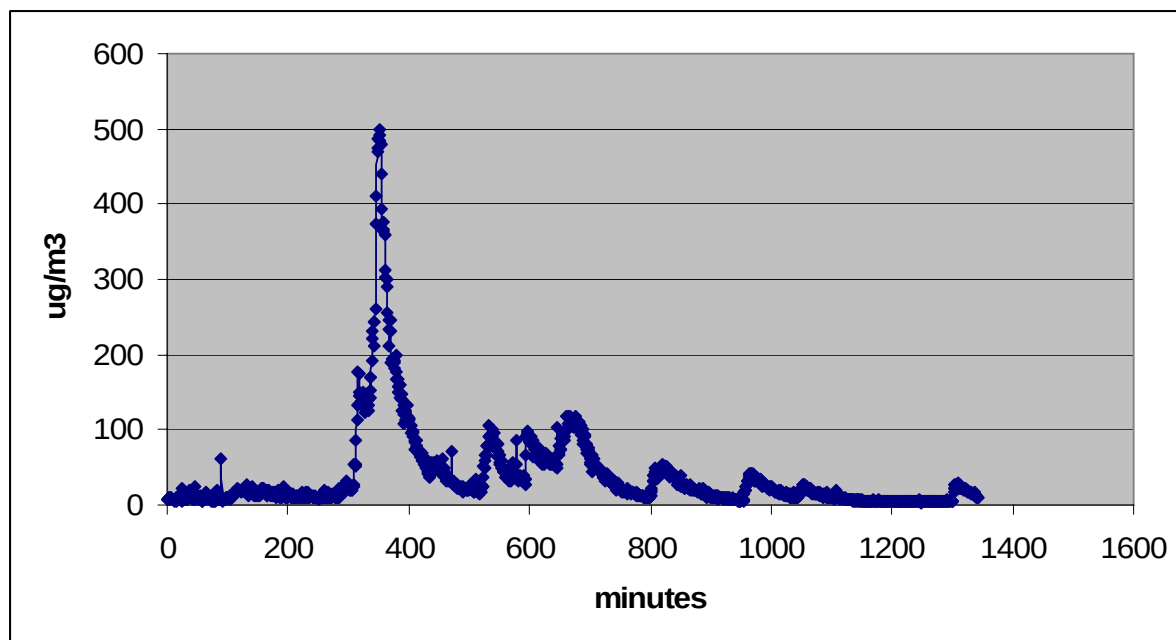
Another source for information on the various activities was the follow-up questionnaire. This questionnaire specifically addressed the times and duration of any cooking, cleaning, and grooming activity as well as other important events. This data was collected using a computer assisted technician administered form. The form was programmed using Visual Basic within ACCESS. The data tables that contained times of the events of interest were converted subsequently into SAS.

This information was stored at the participant/event level with the start times, end times and event type. This information was overlaid with that of the activity diary. If there was an event in either data source, then that time period was set to be an event.

Once all the data were inputted into SAS, the data needed to be further transformed prior to statistical analysis.

LOCATING THE PEAKS

Particulate matter (PM) mass concentrations can change dramatically, especially when a localized source is present. An event will cause there to be an increased level of PM in the air. Conversely, the PM can decrease quickly if there is an active process removing the source. The time that it takes for the PM level mass concentration to return to baseline varies depending upon many factors but often an exponential decay is observed. Peaks vary by both height (maximum concentration) as well as length of existence (source strength). An example of the plot follows:



Most integrating programs used to locate and measure peaks assume a symmetric peak shape. Peaks are often identified by a change in the slope value of the resulting curve as the response grows above the baseline (peak start) and subsequently returns to a baseline value (peak stop). The resulting integrated area under the curve is then calculated using an established algorithm and a mass concentration is determined.

The electronic recording of the data trace is affected not only by the true response from a change in PM concentration, but other events inherent to such instruments (noise, spurious spikes, momentary baseline shifts, etc.)

To overcome these problems, a program was written to locate each peak and generate an integrated value for its response area. Using the macro programming language within SAS, we relied on user defined inclusion and exclusion criteria. Thus, the requirements for defining a peak could be adjusted for different situations. The three criteria were: difference threshold (increase above the baseline), minimum peak height requirement (mass concentration), and the level under which all peaks were declared to be finished. The difference threshold looked for a minimum concentration difference between two consecutive points in order for a start of a peak to be considered. Although the program detects a peak, it may not reach a level that was above its recognized limit of detection. The minimum peak height requirement was used to both remove all participants who had no potential peaks and to re-define small peaks as non-peaks. Finally, the level under which all peaks were declared to be finished was used only as an means to cut off elongated tails having little influence on the integrated area.

PROGRAM DETAILS

Hundreds of thousands of data records were processed. The first step was to limit the data to those participants with at least one value that was greater than the minimum peak height requirement. The simplest way to determine if any value for a participant was above the threshold was to check if the maximum value was above the threshold. Using a proc means, we generated a dataset with only participants with the maximum value above the threshold. We also removed any missing observations so that there would be no calculations using missing data.

```
proc means data=&in_file noprint;
    class pid;
    var mie;
    output out = maxs(where=(max > &peakreq & not(missing(pid))) keep=pid
    max) max=max;
run;

data tempmie;
    merge &in_file maxs(keep = pid in=keepit);
    by pid;
```

```

        if keepit & not(missing(mie));
run;

proc sort data=tempmie;
    by pid dt ;
run;

```

This program is a state dependent program. Information concerning previous observations was critical in understanding the current observation. The retain statement made it possible to retain several variables for consecutive records. These variables included key indicator variables, as well as information about the peaks. Therefore it was important to remember to re-initialize the variables when processing a new participant. By using a "by" statement, SAS automatically generated indicator variables that were used to locate the first and last record associated with the each "by" variable. The other tool we used to obtain information concerning previous records was the lag function. The lag function was used to calculate a running average as well as a localized baseline value.

```

retain stcount peaknum peak ending peaktime maxinpeak baseline starttime 0 ;

array bs(4) b1 b2 b3 b4;

b1 = lag(mie);
b2 = lag2(mie);
b3 = lag3(mie);
b4 = lag4(mie);

smoothed = mean(mie,b1,b2);

prior = lag(smoothed);
ptime = lag(dt);

lastend=lag(ending);

if first.pid then do;
    peaknum = 0;
    peak=0;
    ending = 0;
    smoothed = mie;
    prior = .;
    ptime = .;
    maxinpeak = .;
    baseline = mie;
    lastend = 0;
    starttime = 0;
    stcount = 0;
end;

```

Use of the lag function is tricky and using it conditionally would have been inappropriate. A conditional lag returns not the value of the previous record but the value of the previous function call. Therefore, we do not conditionally set the values using the lag functions. Instead, we redefined the variables if the record was among the first few observations for a participant.

One of the uses of the lag function was to smooth the data. The data were noisy and an inherent feature of PM concentrations in proximity to sources. A running average over three consecutive points was taken to smooth some of the data. The running averages were used in all the calculations except for the baseline. The baseline calculation also relied heavily on lagged values of the data. The baseline was calculated for each peak by averaging the previous four observations.

The slope of the line between consecutive averages was used as a surrogate to the slope of the tangent line. The slope was critical in determining the start and stop of a peak as previously discussed. When missing data were removed from the dataset, gaps were introduced into the time series. Since the time period between the measurements was not constant, the slope was calculated every time as the rise divided by the run.

```
if prior > .z & ptime > .z then slope = 60*(smoothed-prior)/abs(dt-ptime);
```

The dt and ptime in the code above are SAS datetime variables. They are the number of seconds since January 1, 1960. Since the threshold values were given in $\mu\text{g}/\text{m}^3$, the slope was converted into $\mu\text{g}/\text{m}^3/\text{min}$.

To control the processing of the program, conditional state variables were created. The key conditional state indicator variable was PEAK which was a dichotomous variable that indicated if a peak had been found. Once a peak was found, the variable PEAK was set to 1. When the peak was determined to be over, PEAK was set to 0. Whether or not the current observation was during a peak affected the type of processing performed on that record. Once a peak began, we started trying to determine its ending time point. If there was no peak, then we were checking for the start of a peak. Since the end of one peak may be the start of another peak, the check for starting of a peak was done at the end of the data step.

The requirements for a start of a peak were straightforward and composed of 3 conditions: the slope was greater than 5, the observed mass concentration difference was greater than a set threshold, and the ratio between consecutive observations was larger than 0.1. The range of PM mass concentration values was large, thus a single condition was not sufficient. Very large values could have had a small ratio. If only the slope or difference were used, a data section might incorrectly be marked as a peak. Small values could have had a large ratio but a small difference. Once the start of a peak was found, the variables that described the peak were initialized. These variables included the baseline, maximum value within the peak, and the peak number. The PEAK and ENDING variables that were used in the program to direct the flow of the processing were also initialized. Code for location and initializing the start of a peak is below:

```
if peak = 0 & stcount > 3 & (smoothed-prior) > &threshold & round(ratio,.01) >= .1
& round(slope,.1) >= 5 then do;
    maxinpeak = smoothed;
    if stcount < 5 then do;
        do i=1 to 5-stcount;
            bs(5-i)=.;
        end;
    end;

    baseline = mean(b1,b2,b3,b4);
    starttime = dt;
    peak = 1;
    peaknum = peaknum+1;
    ending = 0;
end;
```

If a record was marked as being during a peak, then different processing was done. First, variables that described the peak were captured. These variables included incrementing peak duration and possible maximum value observed within a peak. The most important thing was to check to see if the peak was ending. This was also the most difficult due to the exponential decay of most data traces.

The decline is slow and there might be occasional spikes due to the measurement instrument and inherent noise. Also, one peak can start in the middle of the decline of a previous peak due to the introduction of a new PM source. The basic criterion was the response values getting close to the baseline. While, the baselines were not always constant, they were treated as a constant for each peak. The criteria for declaring completion of a peak was determined over consecutive points. Once 6 consecutive points met the criteria for being a peak completion, the peak was declared to be over. There were three conditions that would cause a peak to be declared immediately over: (1) a sudden spike occurred, (2) the mass concentration value fell below minimum value, and (3) the value is less than 75% of the baseline value.

```
if abs(slope) <= 1 & .75*(maxinpeak-baseline)>= smoothed & round(ratio,.001) <=
.015 then ending = ending +1;
else if round(ratio,.01) < .01 & round(abs(smoothed-baseline) /baseline,0.01)
<=0.05 then ending=ending +1;
else if lastend > 4 & round(ratio,.01) < .02 then ending = ending + 1;
else if lastend > 4 then ending = ending-1;
else ending = 0;
if pslope < 0 & slope > 1 & abs(smoothed-baseline) < 10 then ending = 8;
if smoothed < &minlevel then ending = 9;
```

```

if ending > 6 | smoothed < .75 * baseline then do;
    peak = 0;
    ending = 0;
end;

```

After all possible peaks were found, some further assessment of the peaks was done. A peak had to last more than 5 minutes and it had to reach a minimum mass concentration level. To check these requirements, we needed to summarize over each entire peak separately. The maximum value observed within a peak and the ending time of the peak was in the last record for each peak. A dataset was created with just the last record for each peak. This information was combined with all records for a peak using proc sql. The total time and the maximum values were then compared to the associated minimum requirements. Some of the peaks that were found failed to meet these requirements. These were subsequently reclassified as non-peaks. New numbers were generated to enumerate the remaining peaks.

```

proc sort data=temp2(where=(peak=1)) out=peaks;
    by pid peaknum dt;
run;

data tempmaxs;
    set peaks;
    by pid peaknum;
    if last.peaknum;
    keep pid peaknum maxinpeak endtime;
run;

proc sql;
    create table maxs as
    select a.pid, a.dt, a.peaknum, b.maxinpeak, b.endtime
    from peaks as a, tempmaxs as b
    where a.pid = b.pid & a.peaknum=b.peaknum
    order by pid, peaknum, dt;
quit;
run;

data out_file(rename=(finalnumber=peaknum) drop = peaknum);
    merge peaks(drop=maxinpeak endtime) maxs;
    by pid peaknum dt;

    retain newnum 0;

    totaltime =(endtime-starttime)/60;

    if maxinpeak < &peakreq | totaltime < 5 then peak = 0;

    peaknum    = peaknum*peak;
    endtime    = endtime*peak;
    starttime  = starttime*peak;
    totaltime  = totaltime*peak;

    if peak = 0 then baseline = .;

    if first.pid then newnum=0;
    if first.peaknum & peak = 1 then newnum = newnum +1;
    finalnumber = newnum*peak;

run;

```

The final step for each peak was to calculate estimated area under the curve for use in human exposure analyses. The area under the curve was estimated by summing the product of the time interval and the concentration value for all observations within a peak. Although these calculations could have been done in previous datasteps, the programming was cleaner when only one major process was performed per step. Since the information needed to do the calculations was saved, the additional datastep did not cause that much additional work. In addition, by reducing the peaks from the list of suspected peaks to known peaks the processing was further reduced.

```
data area maxs;
    set out_file(where=(peak=1));
    by pid peaknum;

    area = smoothed*(dt-ptime)/60;
    output area;
    if last.peaknum then output maxs;

run;

proc means data=area noprint;
    class pid peaknum;
    var area;
    output out=areas(where=(not(missing(pid)) & not(missing(peaknum)))) sum=area;
run;

data &out_file;
    merge areas(keep = pid peaknum area)
           maxs (keep = pid peaknum starttime endtime maxinpeak baseline);
    by pid peaknum;
run;
```

CONCLUSIONS

The data needed to perform comparisons among exposure and activities came from a multitude of sources. Each data source was converted into SAS using various techniques. One of the most difficult and time consuming tasks was processing the activity data where open text activity descriptions were converted into a series of dichotomous variables. The willingness to explore alternative methods of re-structuring the data was helpful in getting the job done. A variety of tools both within SAS and outside of SAS simplified the task. This effort indicates that looking at various electronic methods of collecting the information, like using personal digital assistant, web forms, or automatic telephone surveys, would greatly reduce the work needed to prepare the data for analysis. The problem with these approaches is possible loss of information due to not asking the correct question. In addition, there is more up-front work in designing and programming the data capture instrument.

We created a state dependent SAS program to locate the peaks in the MIE data over time. State dependent programs require information concerning the previous state of the data. To save information from one record to the next, the retain statement and the lag function were used. The starts of the peaks were easily found and were determined by comparing the previous value to the current value. The more complicated element was defining where the peaks concluded. A series of checks were performed to determine the completion of a peak.

Once the peaks were identified, summary values were calculated for use in further analysis. These summary values included the area under the curve, maximum value within a peak and the peak duration. The output from this program was combined with the human activity information and published (Wallace, 2006).

COMPLETE CODE

```
*****;
***  Program Name:    findpeaks.sas
***  Author:  Carry Croghan
***  Version:    1
***
***  Purpose  Define peaks programatically
***
```

```

*****;
%macro findpeaks(threshold = 7, peakreq = 30, minlevel = 15, in_file=small,
  out_file=out_file);

proc means data=&in_file noprint;
  class pid;
  var mie;
  output out = maxs(where=(max > &peakreq & not(missing(pid))) keep=pid max) max=max;
run;

data tempmie;
  merge &in_file maxs(keep = pid in=keepit);
  by pid;

  if keepit & not(missing(mie));
run;

proc sort data=tempmie;
  by pid dt ;
run;
*****;

data temp2;
  set tempmie;
  by pid dt ;

  ndate = datepart(dt);

  retain stcount peaknum peak ending peakttime maxinpeak baseline starttime 0 ;

  array bs(4) b1 b2 b3 b4;

  b1 = lag(mie);
  b2 = lag2(mie);
  b3 = lag3(mie);
  b4 = lag4(mie);

  smoothed = mean(mie,b1,b2);

  prior = lag(smoothed);
  ptime = lag(dt);

  lastend=lag(ending);

  if first.pid then do;
    peaknum = 0;
    peak=0;
    ending = 0;
    smoothed = mie;
    prior = .;
    ptime = .;
    maxinpeak = .;
    baseline = mie;
    lastend = 0;
    starttime = 0;
    stcount = 0;
  end;

  if stcount < 6 then stcount=stcount+1;

  if prior > .z & ptime > .z then slope = 60*(smoothed-prior)/abs(dt-ptime);

  pslope = lag(slope);

```

```

if prior > 0 then ratio = abs(smoothed-prior)/prior;

if peak = 1 then do;
    maxinpeak = max(smoothed, maxinpeak);
    peakttime = peakttime +1;
    endtime= dt;
    if abs(slope) <= 1 & .75*(maxinpeak-baseline)>= smoothed & round(ratio,.001) <=
.015 then
        ending = ending +1;

    else if round(ratio,.01) < .01 & round(abs(smoothed-baseline)/baseline,0.01)
<=0.05 then ending=ending +1;
    else if lastend > 4 & round(ratio,.01) < .02 then ending = ending + 1;
    else if lastend > 4 then ending = ending-1;
    else ending = 0;

    if pslope < 0 & slope > 1 & abs(smoothed-baseline) < 10 then ending = 8;
    if smoothed < &minlevel then ending = 9;

end;

if ending > 6 | smoothed < .75 * baseline then do;
    peak = 0;
    ending = 0;
end;

if peak = 0 & stcount > 3 & (smoothed-prior) > &threshold & round(ratio,.01) >= .1
& round(slope,.1) >= 5 then do;
    maxinpeak = smoothed;
    if stcount < 5 then do;
        put pid= dt= stcount= smoothed= b1= b2 = b3 = b4 = ;
        do i=1 to 5-stcount;
            bs(5-i)=.;
        end;
        put pid= dt= b1= b2 = b3 = b4 = ;
    end;

    baseline = mean(b1,b2,b3,b4);
    starttime = dt;
    peak = 1;
    peaknum = peaknum+1;
    ending = 0;
end;
run;

*****;

proc sort data=temp2(where=(peak=1)) out=peaks;
    by pid peaknum dt;
run;

data tempmaxs;
    set peaks;
    by pid peaknum;
    if last.peaknum;
    keep pid peaknum maxinpeak endtime;
run;
*****;

proc sql;
    create table maxs as
    select a.pid, a.dt, a.peaknum, b.maxinpeak, b.endtime

```

```

        from peaks as a, tempmaxs as b
        where a.pid = b.pid & a.peaknum=b.peaknum
        order by pid, peaknum, dt;
quit;
run;

data out_file(rename=(finalnumber=peaknum) drop = peaknum);
    merge peaks(drop=maxinpeak endtime) maxs;
    by pid peaknum dt;

    retain newnum 0;

    totaltime =(endtime-starttime)/60;

    if maxinpeak < &peakreq | totaltime < 5 then peak = 0;

    peaknum    = peaknum*peak;
    endtime    = endtime*peak;
    starttime  = starttime*peak;
    totaltime  = totaltime*peak;

    if peak = 0 then baseline = .;

    if first.pid then newnum=0;
    if first.peaknum & peak = 1 then newnum = newnum +1;
    finalnumber = newnum*peak;

run;

*****;

proc sort data=out_file;
    by pid dt;
run;

proc print data=out_file(obs=15);
run;

data area1 maxs;
    set out_file(where=(peak=1));
    by pid peaknum;

    area = smoothed*(dt-ptime)/60;
    output area1;
    if last.peaknum then output maxs;

run;

proc means data=area1 noprint;
    class pid peaknum;
    var area;
    output out=areas(where=(not(missing(pid)) & not(missing(peaknum)))) sum=area;
run;

data &out_file(keep = pid peaknum starttime endtime area maxinpeak);
    merge areas(keep = pid peaknum area)
        maxs(keep=pid peaknum starttime endtime maxinpeak);
    by pid peaknum;
run;

%mend;

*****;

```

*** end of program;
***** ,

REFERENCES

Aster, R., *Professional SAS Programmer's Pocket Reference*, 5th edition, Paoli, PA: Breakfast Communication Corporation, 2004.

Wallace, L., Williams, R., Rea, A., Croghan, C., "Continuous weeklong measurements of personal exposures and indoor concentrations of fine particles for 37 health-impaired North Carolina residents for up to four seasons", *Atmospheric Environment* 40, 2006.

Wikipedia contributors, "AWK programming language," *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/w/index.php?title=AWK_programming_language&oldid=59649913 (accessed June 21, 2006).

DISCLAIMER

The United States Environmental Protection Agency through its Office of Research and Development funded and managed the research described here. It has been subjected to Agency administrative review and approved for publication. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

CONTACT INFORMATION

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

Carry Croghan
U.S. Environmental Protection Agency
MD-E205-04, 109 TW Alexander Drive
Research Triangle Park, NC 27709
Work Phone: 919 541 3184
Fax: 919 541 0905
Email: Croghan.Carry@epa.gov

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.