ABSTRACT
Have you ever been frustrated trying to get your output to break pages correctly? If so, the macro presented in this paper will appeal to you. The PAGENUMS macro allows for multiple columns of wrapped text, keeping data grouped together on the page and forcing page breaks.

MOTIVATION
Defining the page numbering for an adverse event listing was the motivation for this macro. The listing required 1) multiple columns of information that contained wrapped text, 2) displaying a subject's adverse events on one page (when possible), and 3) starting each treatment on a new page.

SPECIFIC EXAMPLE
The following is an example of a data listing that will be used throughout this paper.

Adverse Event Listing
Page 1 of XX
Treatment #1

<table>
<thead>
<tr>
<th>SUB</th>
<th>BODY SYSTEM</th>
<th>PREFERRED TERM</th>
<th>VERBATIM TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Body System #1</td>
<td>Preferred Term #1</td>
<td>Verbatim Text #1</td>
</tr>
<tr>
<td>002</td>
<td>Body System #1</td>
<td>Preferred Term #1</td>
<td>Verbatim Text #1</td>
</tr>
</tbody>
</table>

-really long text

<table>
<thead>
<tr>
<th>SUB</th>
<th>BODY SYSTEM</th>
<th>PREFERRED TERM</th>
<th>VERBATIM TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Body System #2</td>
<td>Pref. Term #2</td>
<td>Verbatim Text #2</td>
</tr>
</tbody>
</table>

PREPARE THE INPUT DATA
Before the PAGENUMS macro can be applied, the variables that are going to be displayed in columns that wrap in the output need to have a character inserted to identify where the wrap will occur. For this example a caret (^) will be used and the data for subject 001 is displayed below.

AEDATA data set:
TREAT=1
SUBID='001'
BODYSYS='Body System #1'
PREFTERM='Preferred Term #1'
VERBTEXT='Verbatim Text #1'

MACRO VARIABLES USED
There are several macro variables used in the PAGENUMS macro that need to be defined in the call. An explanation of these macro variables is provided below.

INDATA: Name of input data set.
OUTDATA: Name of output data set.
UNIQUE: Variables needed to uniquely identify an observation.
WRAPVARS*: Variables that can have wrapped text.
IDWRAP: Character that identifies where a wrap occurs.
KEEPONPG: Variables that identify what is trying to be kept together on the page.
KEEPID: Variables needed to merge the KEEPONPG variable.
SKIPLINE: Number of lines to be skipped between unique observations.
XTRFIRST: Number of additional lines needed for the first observation of the KEEPONPG variable.
LINESAVL: Number of lines available per page - excluding those needed for the header and footer.

* Identifies macro variables that can be left null if not applicable.

THE PAGENUMS MACRO
The overall goal of the PAGENUMS macro is to assign a page number to each observation in the data set keeping as much KEEPONPG information together on the same page as possible. The macro adds the variable PAGENO to the data set specified by the INDATA macro parameter.

The code for the PAGENUMS macro is available in Appendix 1. The following is a description of each step used in the macro.

Step 1. Count the number of lines per observation.
The first step is to count the number of lines used in each column containing wrapped text for each observation and output the maximum. This is automated by using the dim function as part of a do loop that utilizes arrays. The dim function counts the number of variables containing wrapped text. This count is used
as the max for a loop that counts the number of lines needed per variable. The number of lines needed is determined by counting the number of wrap identifying characters (in this example carets) in each text string and adding one. Only the maximum count is kept for each observation.

**Step 1a. Add additional lines.**
The number of lines specified by the SKIPLINE and XTRFIRST macro variables are added to the count of the maximum number of lines per observation.

**Step 2. Count the number of lines per KEEPONPG variable.**
Once the maximum number of lines necessary per observation has been determined, the next step is to count the number of lines that will be needed for each set of observations that you want to keep together. For the adverse event listing example, the goal is to keep all the adverse events for each subject together. The counts determined in this step will be set aside in a data set which also contains the KEEPID variables.

**Step 3. Merge data sets.**
The data set created in Step 2 is now merged onto the rest of the data by the KEEPID variables.

**Step 4. Determine the page breaks.**
This is a long step since there are several situations to take into account in the programming. The purpose of this step is to assign a page number to each observation in the data set keeping as much KEEPONPG information together on the same page as possible.

**Step 4a. Forced page breaks.**
Forced page breaks occur when a new value of the FORCEBRK macro variable is encountered.

**Step 4b. KEEPONPG variable that needs only 1 page.**
If the count determined in Step 2 is less than or equal to the lines available on the page, then all the KEEPONPG information can be kept together on one page. The macro checks to see if there is enough room available on the current page to display all the KEEPONPG information. If so, the current page is assigned. If not, the macro assigns the next page to be used.

**Step 4c. KEEPONPG variable needs more than 1 page.**
If the count determined in Step 2 is greater than the lines available on the page, then the KEEPONPG information can not be kept together on one page. For this situation, the macro will put as many observations as possible on the current page and then assign subsequent pages as needed.

**CREATE SPECIFIC EXAMPLE OUTPUT**
The output data set created by the PAGENUMS macro can be used with PROC REPORT to create output files. The following code was used to create the example adverse event listing displayed at the beginning of this paper.

```sas
%let LONGLINE=%sysfunc(repeat(,, %eval(43)));
proc report data=OUT split='\^' spacing=2;
  columns PAGENO TREAT SUBID BODYSYS PREFTERM VERBTEXT;
  define PAGENO / order noprint;
  define TREAT / order noprint;
  define SUBID / order left width=3 'SUB'ID'--' spacing=0;
  define BODYSYS / order left flow width=10 'BODY'SYSTEM'--';
  define PREFTERM / order left flow width=13 'PREFERRED'TERM'--';
break after VERBTEXT / skip;
compute after PAGENO / skip;
  line &LONGLINE;
endcomp;
break after PAGENO / page;
by PAGENO TREAT;
title1 'Adverse Event Listing';
title2 'Page 1 of XX';
title3 ' ';
title4 "Treatment #byval(TREAT)";
title5 &LONGLINE;
run;
```

**CONCLUSIONS**
The PAGENUMS macro was designed to be robust under many circumstances. I have found this to be true. The macro is extremely helpful in determining the page counts for many different kinds of output. The only shortcoming I have found is that it does not take into account multiple levels of information that should be kept together on a page.

**ACKNOWLEDGMENTS**
Thanks to Brian Thornton for his encouragement and assistance in creating the PAGENUMS macro.

**CONTACT INFORMATION**
Lara E.H. Guttadauro, MS
Kendle International Inc.
1200 Carew Tower
441 Vine Street
Cincinnati, OH  45202
Work Phone: (513) 763-1363
Email: guttadauro.lara@kendle.com
/*
  APPENDIX 1: THE PAGENUMS MACRO
  
  | PROGRAM NAME: PAGENUMS.SAS          |
  | PROGRAMMER: Lara Guttadauro         |
  | SOFTWARE AND VERSION#: SAS 8.02     |
  | PURPOSE: Create macro that will determine page breaks. |
  
  **NOTE: A character needs to be included in the variables that will wrap that will identify **;
  ** when the text will wrap to the next line. **;

  $macro PAGENUMS (INDATA=, /* Name of input data set */
  OUTDATA=, /* Name of output data set */
  UNIQUE=, /* Identifies unique observations */
  WRAPVARS=, /* Variables that can potentially have wrapped text */
  IDWRAP=, /* Identifier for text going to next line */
  KEEPONPG=, /* Var that ids what is trying to be kept together on 1 page */
  KEEPID=, /* Variables needed to merge by the KEEPONPG variable */
  XTRFIRST=, /* # of additional lines needed for first obs for KEEPONPG var */
  SKIPLINE=, /* # of lines to be skipped between unique observations */
  LINESAVL=, /* # of lines available per page - excluding header & footer */
  FORCEBRK= /* Var that needs to be forced to break on a new page */);

  %** Sort the incoming data set **;
  proc sort data=&INDATA out=MDATA1;
  by &UNIQUE;
  run;

  %** Count the maximum number of lines per observation **;
  data MDATA2;
  set MDATA1;
  by &UNIQUE;
  %if &WRAPVARS ne %then %do;
    array WRAP{*} &WRAPVARS;
    MAXLINES=0;
    do i=1 to dim(WRAP);
      MAXLINES=max(MAXLINES,
        length(compress(WRAP{i}))-length(compress(WRAP{i}," &IDWRAP"))+1);
    end;
    %end;
  %else %do;
    MAXLINES=1;
    %end;

  %** Add the line skips and the additional lines for the first KEEPONPG variable **;
  if first.&KEEPONPG then MAXLINES=MAXLINES+&SKIPLINE+&XTRFIRST;
  else MAXLINES=MAXLINES+&SKIPLINE;
  run;

  %** Count the number of lines needed for each KEEPONPG variable **;
  data MDATA3 (keep=&KEEPMID KEEPLINE);
  set MDATA2;
  by &UNIQUE;
  retain KEEPLINE 0;
  if first.&KEEPONPG then KEEPLINE=MAXLINES;
  else KEEPLINE=KEEPLINE+MAXLINES;
  if last.&KEEPONPG;
  run;

  %** Merge the maximum number of lines per KEEPONPG variable onto rest of data **;
  data &OUTDATA (drop=LINEONPG CHECKFIT KEEPLINE MAXLINES PAGENEED);
    merge MDATA2 MDATA3;
    by &KEEPMID;
  run;
*/
PAGENEED=ceil(KEEPLINE/&LINESAVL);
%
** Identify the page breaks **;
retain PAGENO 1 LINEONPG 0 CHECKFIT;
%
if &FORCEBRK ne %then %do;
  ** If it is a FORCED BREAK (and not the first one), then go to next page **;
  if first.&FORCEBRK and _n_ ne 1 then do;
    PAGENO=PAGENO+1;
    LINEONPG=0;
  end;
%
** Deal with KEEPONPG variable that only need one page **;
if PAGENEED=1 then do;
  ** Check to see if KEEPONPG variable will fit on the page **;
  if first.&KEEPONPG then CHECKFIT=LINEONPG+KEEPLINE;
    ** YES - Whole KEEPONPG variable fits **;
    if CHECKFIT<=&LINESAVL then LINEONPG=LINEONPG+MAXLINES;
    ** NO - KEEPONPG variable will not fit **;
    if CHECKFIT>&LINESAVL then do;
      PAGENO=PAGENO+1;
      LINEONPG=MAXLINES;
      CHECKFIT=KEEPLINE;
    end;
  end;
%
** Deal with KEEPONPG variable that need more than one page **;
if PAGENEED>1 then do;
  ** Check to see if UNIQUE variable will fit on the page **;
  if first.&KEEPONPG then CHECKFIT=LINEONPG+MAXLINES;
    ** YES - UNIQUE variable fits **;
    if CHECKFIT<=&LINESAVL then LINEONPG=LINEONPG+MAXLINES;
    ** NO - UNIQUE variable does not fit **;
    if CHECKFIT>&LINESAVL then do;
      PAGENO=PAGENO+1;
      LINEONPG=MAXFIRST+MAXLINES;
      CHECKFIT=MAXFIRST+MAXLINES;
    end;
  end;
run;
%mend PAGENUMS;