ABSTRACT
In the pharmaceutical reporting world, page breaks can be a remarkably frustrating experience. Some data are very simple to report, the text variables do not wrap onto the next line and there is a set number of lines you wish to display on each page. However, often this is not the case – adverse events, physical examinations, concomitant medications, or just having to report unscheduled visits in the correct date order without knowing how many there could be. This paper will present you with a range of options for setting up page break variables, solving simple to more advanced problems. All data will be reported using the REPORT procedure but the page break variable will also be useful for other reporting procedures or practices. Intended audience – beginner to intermediate, not limited to any particular operating system, based on SAS® 8.2.

KEYWORDS: page breaks, PROC REPORT, adverse events, counting

INTRODUCTION
When producing tables and listings, presentation is always an important consideration. Yes, it is often easy to produce something that is acceptable but then it can be quite painstaking and time consuming to order and display the data neatly and in an easily understandable format. For any kind of summary table, even the simplest, it is often a good idea to add in a page break variable. When you first write your code the reporting step might run perfectly but when it is subsequently rerun with new data, variables might wrap onto new lines, or wrap onto new pages where you are least expecting it. If you add in a page break variable it will help towards making your code more robust and dependable.

A SIMPLE PAGE BREAK
Let us start with an example of a basic page break for a standard frequency table, presenting groups of data with a consistent number of lines in each group. A good example of this is a physical examination summary table.

By looking at the page size you require and the number of headers and footers you are including in your output you have decided that you can have 12 lines of output on each page. Therefore at this point a page variable can be added to your data set.

data page;
  set test;
    p=ceil(_n_/12);
run;
Use the CEIL function and the automatic _n_ variable (observation number) found in every data set to increment the value of p after every twelve records.

PAGE data set: partial print out

<table>
<thead>
<tr>
<th>vis</th>
<th>param</th>
<th>scale</th>
<th>scale1</th>
<th>scale2</th>
<th>...</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Abdomen</td>
<td>Not Done</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Abdomen</td>
<td>Normal</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Abdomen</td>
<td>Abnormal CS</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Abdomen</td>
<td>Abnormal NCS</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Cardiovascular</td>
<td>Not Done</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Cardiovascular</td>
<td>Normal</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Cardiovascular</td>
<td>Abnormal CS</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Cardiovascular</td>
<td>Abnormal NCS</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>General Appearance</td>
<td>Not Done</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>General Appearance</td>
<td>Normal</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>General Appearance</td>
<td>Abnormal CS</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>General Appearance</td>
<td>Abnormal NCS</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>HEENT</td>
<td>Not Done</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

Now, in PROC REPORT, you can order by the variable p, break after it and skip a page every time it changes value.

```sql
proc report data=page headline headskip nowd split='~' missing formchar(2)=';
column ('__' p vis param scale ('Treatment 1~__' scale1-scale4) ('Treatment 2~__' scale5-scale8));
define p        / order noprint;
define vis      / order noprint;
define param    / order order=data 'Parameter'    width=30 left flow;
define scale    / order order=data ' '            width=15 left flow;
define scale1   /                  'Not Done'     width=10 left;
define scale2   /                  'Normal'       width=10 left;
define scale3   /                  'Abnormal CS'  width=10 left;
define scale4   /                  'Abnormal NCS' width=10 left;
...  
define scale8   /                  'Abnormal NCS' width=10 left;
break after p / page;
break after param / skip;
run;
```

Example 1: partial output for a simple page break, 12 lines on each page

----------------------------------
<table>
<thead>
<tr>
<th>General Appearance</th>
<th>Not Done</th>
<th>Normal</th>
<th>Abnormal CS</th>
<th>Abnormal NCS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
</tr>
</tbody>
</table>
----------------------------------
Example 1: simple page break, 12 lines on each page, page 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Treatment 1</th>
<th>Treatment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not Done</td>
<td>Normal</td>
</tr>
<tr>
<td></td>
<td>Abnormal CS</td>
<td>Abnormal NCS</td>
</tr>
<tr>
<td>HEENT</td>
<td>Not Done</td>
<td>0 (0.0%)</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>0 (0.0%)</td>
</tr>
</tbody>
</table>
```
MORE COMPLEX PAGE BREAKS

Now let us have a look at some more complex page breaks. The second example is a summary statistics table, presented at baseline and at a subsequently unknown number of visits, including change from baseline; that is, presenting groups of data with a consistent number of lines in each group, but at an unknown number of visits where certain visits should be grouped together. A good example of this is an ECG summary table, where unscheduled visits can occur in between any scheduled visits. The number and placement of unscheduled visits could easily change with each new data extract.

TEST data set: partial print out

<table>
<thead>
<tr>
<th>vis</th>
<th>cfb</th>
<th>param</th>
<th>group1</th>
<th>...</th>
<th>stats</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>No</td>
<td>Actual QT Interval (msec)</td>
<td>0.0</td>
<td></td>
<td>Median</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Actual QT Interval (msec)</td>
<td>0,0</td>
<td></td>
<td>Min,Max</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Corrected QT (QTcB) Interval (msec)</td>
<td>0.0 (0.00)</td>
<td>Mean (SD)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Corrected QT (QTcB) Interval (msec)</td>
<td>0.0</td>
<td></td>
<td>Median</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Corrected QT (QTcB) Interval (msec)</td>
<td>0,0</td>
<td></td>
<td>Min,Max</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Corrected QT (QTcF) Interval (msec)</td>
<td>0</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Corrected QT (QTcF) Interval (msec)</td>
<td>0.0 (0.00)</td>
<td>Mean (SD)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Corrected QT (QTcF) Interval (msec)</td>
<td>0.0</td>
<td></td>
<td>Median</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>Ventricular Rate (beats/min)</td>
<td>0</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>Ventricular Rate (beats/min)</td>
<td>0.0 (0.00)</td>
<td>Mean (SD)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>Ventricular Rate (beats/min)</td>
<td>0.0</td>
<td></td>
<td>Median</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>Ventricular Rate (beats/min)</td>
<td>0,0</td>
<td></td>
<td>Min,Max</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>PR Interval (msec)</td>
<td>0</td>
<td></td>
<td>N</td>
</tr>
</tbody>
</table>

Each new parameter needs to be displayed on a new page; baseline, visit 2 and change from baseline at visit 2 need to be displayed on the same page and then for subsequent visits the actual values and change from baseline values need to be on the same page per visit. At this point, you can add a page variable into your data set to aid PROC REPORT to consistently break the output in the desired places.

```
proc sort data=test out=test1;
    by param vis cfb;
run;

data pall;
    set test1;
    by param vis cfb stats notsorted;
    if vis not in (1 2) then do;
        if first.vis then pg+1;
    end;
    if first.param then pg+1;
run;

proc report data=pall headline nowd spacing=2 headskip split='~' formchar(2)='_'
    missing;
    column ('__' pg param vis cfb stats group:);
    define pg         / order noprint;
    define param      / order "Parameter"            left width=50 flow;
    define vis        / order order=internal "Visit" left width=15 format=4.1;
    define cfb        / order "Change from-Baseline" left width=12;
    define stats      /       " "                    left width=16 order=data;
    define group1     /       "Treatment 1"          left width=15;
    define group2     /       "Treatment 2"          left width=15;
    define group3     /       "Total"                left width=15;
    break after cfb /skip;
    break after pg / page;
run;
```
Example 2: partial output using a page break for data with consistent group size but at unknown visits and visit types

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Visit</th>
<th>Change from Baseline</th>
<th>Treatment 1</th>
<th>Treatment 2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual OT Interval (sec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>No</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mean (SD)</td>
<td>0.0 (0.0)</td>
<td>0.0 (0.0)</td>
<td>0.0 (0.0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Median</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min, Max</td>
<td>0.0, 0.0</td>
<td>0.0, 0.0</td>
<td>0.0, 0.0</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td>No</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mean (SD)</td>
<td>0.0 (0.0)</td>
<td>0.0 (0.0)</td>
<td>0.0 (0.0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Median</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min, Max</td>
<td>0.0, 0.0</td>
<td>0.0, 0.0</td>
<td>0.0, 0.0</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td></td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mean (SD)</td>
<td>0.0 (0.0)</td>
<td>0.0 (0.0)</td>
<td>0.0 (0.0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Median</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min, Max</td>
<td>0.0, 0.0</td>
<td>0.0, 0.0</td>
<td>0.0, 0.0</td>
</tr>
</tbody>
</table>

**PAGE BREAKS FOR DATA WITH UNKNOWN BLOCK SIZE – INCLUDING ADVERSE EVENTS, CONCOMITANT MEDICATIONS**

By far the most complex type of table to report with consistent page breaks are tables with blocks of data that need to be grouped together but where the size of the blocks vary. A common example of this is reporting adverse events, concomitant medications, prior medications and medical histories. Often adverse event tables can look messy and disjointed if events from the same system organ class flow onto separate pages. It is quite a challenge to write a robust piece of code to break the pages correctly, no matter how the data might change in the future.

Here is an example data set holding adverse event information that has been formatted for table output.

**TEST data set: partial print out**

```
Obs  soc_term
1    Any Adverse Events
2    Cardiac disorders
3    Cardiac disorders
4    Ear and labyrinth disorders
5    Ear and labyrinth disorders
6    Gastrointestinal disorders
7    Gastrointestinal disorders
8    Gastrointestinal disorders
9    Gastrointestinal disorders
10   Gastrointestinal disorders
11   Gastrointestinal disorders
12   General disorders and administration site conditions
13   General disorders and administration site conditions
14   General disorders and administration site conditions
15   General disorders and administration site conditions

Obs  list                  cat  percl  ...  evenl  ...
1    Any Adverse Events    1    9 (69.2) 36   ...
2    Cardiac disorders     1    0 (0.0)  0    ...
3    Angina pectoris       2    0 (0.0)  0    ...
4    Ear and labyrinth disorders 1    1 (7.7)  1    ...
5    Vertigo               2    1 (7.7)  1    ...
6    Gastrointestinal disorders 1    5 (38.5) 6    ...
7    Abdominal pain        2    1 (7.7)  1    ...
8    Constipation          2    2 (15.4) 2    ...
```
If these data are reported using a PROC REPORT step without a page break variable they will not group into system organ class blocks but rather PROC REPORT will report as many lines as possible before running onto a new page. It is possible to skip a line or skip a page after each system organ class, but this will mean system organ classes still running over pages or system organ class groups with only two lines taking up a whole page of output. By running the data through the following macro a page break variable can be added to the data set before the PROC REPORT step.

**MACRO ASSUMPTIONS**

Data are pre-formatted into SOC_TERM (system organ class or for other data, main subgroup), CAT (category - the group heading or a member of that group – for adverse events, system organ class or preferred term) and LIST holds the text to output for each line. For an example, see the print out of the data set TEST above. Sort the data set by SOC_TERM, CAT and LIST. The macro can be easily adapted for different types of data, more or less sub grouping and with modifications the code will fit a number of different situations.

Description of macro variables:

**DSETIN** = the name of the input data set.

**LEN** = the length of the variable LIST in the subsequent PROC REPORT step.

**PG** = the number of usable lines in your report, that is the number of spaces between the top line and bottom line of your page, taking into account titles, footnotes and PROC REPORT headings and spacing. Simply, you can count this on screen after a trial run, or depending on your method of reading in titles and footnotes and the page size, you can add in a programming step to calculate this for you. (This step is not included in this macro).

```%macro pager(dsetin=,len=,pg=);
  /* Set up page variable */
  /* Take account of wrapping text (length of list in proc report) */
  /* Assumes that no lengths of list are more than double this */
  data pager;
    set &dsetin(keep=soc_term cat list);
    by soc_term cat list;
    if first.soc_term then sblock=0;
    if length(list) le &len then sblock+1;
    else if length(list) gt &len then sblock+2;
    if last.soc_term then do;
      sblock+1; /* to take account of line skip in proc report */
      output;
    end;
  run;

  /* Set rolling count */
  /* Sum rolling count and when it reaches page restriction make a new page */
  data pager1(drop=endb sblock list cat);
    retain endb 0;
    set pager;
    by soc_term cat list;
    endb=endb+sblock;
    if (endb gt %eval(&pg+1) gt sblock) or (sblock gt &pg) then do;
      page+1;
    end;
%
```

5
if sblock le &pg then endb=sblock;
else if sblock gt &pg then endb=(sblock-&pg);
end;
run;

/* merge back with all output */
data perc;
merge pager1
   &dsetin;
by soc_term;
run;
%mend pager;
%
 pager(dsetin=perca, len=50, pg=20);  /* example macro call */

PERC data set – partial print out

Obs  soc_term
1    Any Adverse Events
2    Cardiac disorders
3    Cardiac disorders
4    Ear and labyrinth disorders
5    Ear and labyrinth disorders
6    Gastrointestinal disorders
7    Gastrointestinal disorders
8    Gastrointestinal disorders
9    Gastrointestinal disorders
10   Gastrointestinal disorders
11   Gastrointestinal disorders
12   General disorders and administration site conditions
13   General disorders and administration site conditions
14   General disorders and administration site conditions
15   General disorders and administration site conditions

Obs  page   list                                cat    perc1   ... even1 ...
1    0    Any Adverse Events                   1    9 (69.2)      36
2    0    Cardiac disorders                    1    0 ( 0.0)       0
3    0      Angina pectoris                    2    0 ( 0.0)       0
4    0    Ear and labyrinth disorders          1    1 ( 7.7)       1
5    0      Vertigo                            2    1 ( 7.7)       1
6    0    Gastrointestinal disorders           1    5 (38.5)       6
7    0      Abdominal pain                     2    1 ( 7.7)       1
8    0    Constipation                        2    2 (15.4)       2
9    0    Diarrhoea                           2    1 ( 7.7)       1
10   0    Nausea                              2    1 ( 7.7)       1
11   0    Salivary hypersecretion             2    1 ( 7.7)       1
12   1    General disorders and administra...  1    2 (15.4)       4
13   1    Asthenia                            2    1 ( 7.7)       2
14   1    Fatigue                             2    0 ( 0.0)       0
15   1    Lethargy                           2    1 ( 7.7)       1

As you can see above, the new variable PAGE only changes when there is a change of system organ class and another system organ class block will not fit onto the same page, or in the case that one system organ class holds more than a page’s worth of output, when the page is full. Below is an example PROC REPORT step to use with the PERC data set.
Example 3: partial output showing page break for data with unknown block size

<table>
<thead>
<tr>
<th>System Organ Class</th>
<th>Preferred Term</th>
<th>Treatment 1 (N=12)</th>
<th>Treatment 2 (N=17)</th>
<th>Total (N=29)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Number (%) of patients</td>
<td>Number of events</td>
<td>Number (%) of patients</td>
</tr>
<tr>
<td>Any Adverse Events</td>
<td></td>
<td>0 (0.0)</td>
<td>0</td>
<td>1 (3.3)</td>
</tr>
<tr>
<td>Cardiac disorders</td>
<td></td>
<td>0 (0.0)</td>
<td>0</td>
<td>1 (3.3)</td>
</tr>
<tr>
<td>Angina pectoris</td>
<td></td>
<td>0 (0.0)</td>
<td>0</td>
<td>1 (3.3)</td>
</tr>
<tr>
<td>Ear and Labyrinth disorders</td>
<td></td>
<td>1 (7.1)</td>
<td>0 (0.0)</td>
<td>0</td>
</tr>
<tr>
<td>Vertigo</td>
<td></td>
<td>1 (7.1)</td>
<td>1</td>
<td>0 (0.0)</td>
</tr>
<tr>
<td>Abdominal pain</td>
<td></td>
<td>0 (0.0)</td>
<td>0</td>
<td>1 (3.3)</td>
</tr>
<tr>
<td>Constipation</td>
<td></td>
<td>2 (15.4)</td>
<td>2</td>
<td>0 (0.0)</td>
</tr>
<tr>
<td>Diarrhoea</td>
<td></td>
<td>1 (7.1)</td>
<td>1</td>
<td>0 (0.0)</td>
</tr>
<tr>
<td>Nausea</td>
<td></td>
<td>1 (7.1)</td>
<td>1</td>
<td>0 (0.0)</td>
</tr>
<tr>
<td>Gallbladder hyperesthesia</td>
<td></td>
<td>1 (7.1)</td>
<td>1</td>
<td>0 (0.0)</td>
</tr>
</tbody>
</table>

Note: Subjects with more than one event coded to the same preferred term are counted only once.

Note: Percentages are based on the number of subjects in the safety population.

In order for the macro code to be easily reused and adapted for different situations, see the breakdown below detailing each step. The descriptions assume you are reporting adverse event data.

```sas
%macro pager(dsetin=, len=, pg=);

data pager;
  set &dsetin(keep=soc_term cat list);
  by soc_term cat list;

  if first.soc_term then sblock=0;
run;
```

Set your data set containing the formatted data ready to report.

For the first line of each system organ class block set variable SBLOCK (size of block) to zero.
Check the length of the variable LIST (contains the text for the system organ classes and preferred terms), if the length of LIST is less than or equal to the macro variable &LEN (length of the variable LIST in the PROC REPORT step) then increase the value of SBLOCK by 1 (that is, this text will fit onto one line in the PROC REPORT output).

```plaintext
if length(list) le &len then sblock+1;
```

If the length of LIST is greater than &LEN then increase the value of SBLOCK by 2 (although one line in the data set, this text will take up two lines in the PROC REPORT output). If you know that your variable is likely to be 3 (or more times) longer than &LEN, additional ELSE clauses can be added to make SBLOCK+3, +4 and so on.

```plaintext
else if length(list) gt &len then sblock+2;
```

If it is the last observation in the system organ class block then additionally increase the value of SBLOCK by 1 (to take account of the break in the PROC REPORT step that adds a blank line in between system organ class blocks) and print the line to the output data set.

```plaintext
if last.soc_term then do;
    sblock+1;
    output;
end;
run;
```

Start a new DATA step and set the previous data set (holding the block size for each system organ class block). Also, use a RETAIN statement for the new variable ENDB and retain it with the value of zero.

```plaintext
data pager1(drop=endb sblock list cat);
retain endb 0;
set pager;
    by soc_term cat list;
endb=endb+sblock;
```

Start a cumulative count of system organ class block size and set the value of ENDB to this value.

```plaintext
endb=endb+sblock;
```

Check if the current value of ENDB (cumulative sum of the block sizes) is greater than the page size + 1 and check if the page size + 1 is greater than the current value of SBLOCK, or check for the condition where SBLOCK is greater than the page size. If either of these two conditions are true then add 1 onto a variable called PAGE (this is your page break variable).

These steps ensure that if there is no more room on a page to accommodate a whole new block then the page will increase by one. The value of page size + 1 is used because in an earlier assumption 1 was added to block size to take account for the break after system organ class in the PROC REPORT step, however if the block is at the end of a page this extra blank line is not needed. If the whole block size is larger than the page size (a very large system organ class block), then this block will have to be split over two pages.

```plaintext
if (endb gt %eval(&pg+1) gt sblock) or (sblock gt &pg) then do;
    page+1;
```
Within the same DO LOOP, check for the condition where the block size (SBLOCK) is less than or equal to the page size, if this is true set the value of ENDB (the cumulative count of block size) to the current value of SBLOCK. This will ensure that during the next pass of the DATA step a new count is started to create the next page break dependent on how many system organ class blocks can fit onto the next page.

If the block size is greater than the page size (a very large system organ class group) set the value of ENDB to the current value of SBLOCK minus the page size, these are the leftover lines from the large block that will flow onto the new page.

The DATA step processes each observation in turn until the end of the data set is reached.

```sas
if sblock le &pg then endb=sblock;
   else if sblock gt &pg then endb=(sbblock-&pg);
end;
run;
```

The following DATA step merges the data from PAGER1, containing one line per system organ class and the page on which each system organ class block will appear, with the original data set called in the first DATA step of the macro.

```sas
data perc;
  merge pager1
    &dsetin;
  by soc_term;
run;
%mend pager;
```

**CONCLUSION**

Page breaks can be very frustrating, however they can always be solved programmatically. Although specific examples have been given in this paper, the code samples should prompt solutions for a number of problems. For straightforward page breaks a simple arithmetic approach can be taken. For more complex solutions, take into account different types of output and use cumulative counts to group the data. The end result is neat, easily decipherable and professional looking output.

**REFERENCES**

SAS OnlineDoc® documentation for SAS version 8.2.

**ACKNOWLEDGMENTS**

Thanks to colleagues Stephen Hunt and Alex Olea who gave their time to review and give valuable feedback.

**CONTACT INFORMATION**

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

Suzanne Humphreys  
PRA International  
300-730 View Street, Victoria, BC, Canada V8W 3Y7  
Work Phone: 250-483-4404  
Fax: 250-483-4588  
E-mail: HumphreysSuzanne@PRAIntl.com  
Web: www.prainternational.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.