Paper TT12

Effective Strategy to Set Page Breaks for ODS RTF Output

Songtao Jiang, Daniel Boisvert, HCRI, Boston, MA

ABSTRACT

PROC REPORT, combined with output deliver system (ODS), is a very powerful and widely used report generator in SAS®. By allowing Microsoft Word to control the printing process, SAS users lose some of the power to control the layout of the output. This makes it almost impossible to set appropriate page breaks for ODS rtf output. This paper proposes a simple and effective strategy to force correct page breaks. By calculating number of printed rows for each data set observation, the proposed algorithm assigns each observation a page number according to the column widths and page size, which are set by the user. This approach is extremely easy, useful and efficient in day-to-day work for SAS programmers.

Key word: SAS, ODS RTF, PROC REPORT, Listing, Page Break

1. INTRODUCTION

1.1 ODS RTF output

ODS RTF (rich text format) outputs the result with formatting information to an RTF file. The RTF reader interprets, by default the formatted result and renders it. Usually this is done by Microsoft Word [BS] . Since RTF is one of the formats in which the FDA will accept electronic submissions, RTF is often chosen as the primary format of pharmaceutical companies.

1.2 Limitation of ODS RTF

One big limitation is that ODS RTF does not do "vertical measurement" [BS]. This means that SAS RTF does not know where the optimal place is to position each printed item on the page vertically. It leaves the task of page breaking to the RTF interpreter. Thus, the user is no longer in complete control of the page breaking.

In practice, it is straight-forward to set page breaks when each observation only appears on one line. However, many tables or listings are not in this one-observation-on-one-line situation. Since ODS RTF does not do vertical measurement, there are tables or listings that require the SAS programmers to set appropriate page breaks. The length of some data values causes text wrapping within the cells. SAS will not know how many observations will fit on the page. The programmer must manually count the number of lines for each observation and sets page breaks. So far, there is no precise and efficient way to do it automatically in SAS ODS output.

In the paper "Pagination in the ODS" [JK], the author presented an approach, which takes advantage of other programming packages like Microsoft Foundation Class for C++ (MFC) or Java. Technically, it is a very good approach. However, there are a few limitations. First, MFC for C++ is not free software. Second, Java

objects can only be used in SAS 9 via JAVAOBJ data step. Third, not every SAS programmer knows these programming languages, making the codes difficult to maintain.

2. SOLUTION

2.1 Overview

The proposed algorithm is simple. No advanced SAS programming skill is needed. It can be used by any SAS programmer on any system with SAS ODS support. The basic idea for this algorithm is to find the number of lines for each observation. By calculating the length of the string of each given variable of one observation, the algorithm can figure out number of lines for the printed observation, provided with the width of each cell and number of lines on the page.

2.2 The utility text file

The primary part of this algorithm is to figure out the width of each character. In our day-to-day work, we use 8pt “arial” font, which is one standard font for FDA submission. We are using this font for our example. First step is to create an empty Word file, set the margins to 6 inches and set the font to “arial, 8”, and populate each line with the same character. One line presents one kind character, as shown in Figure 1.

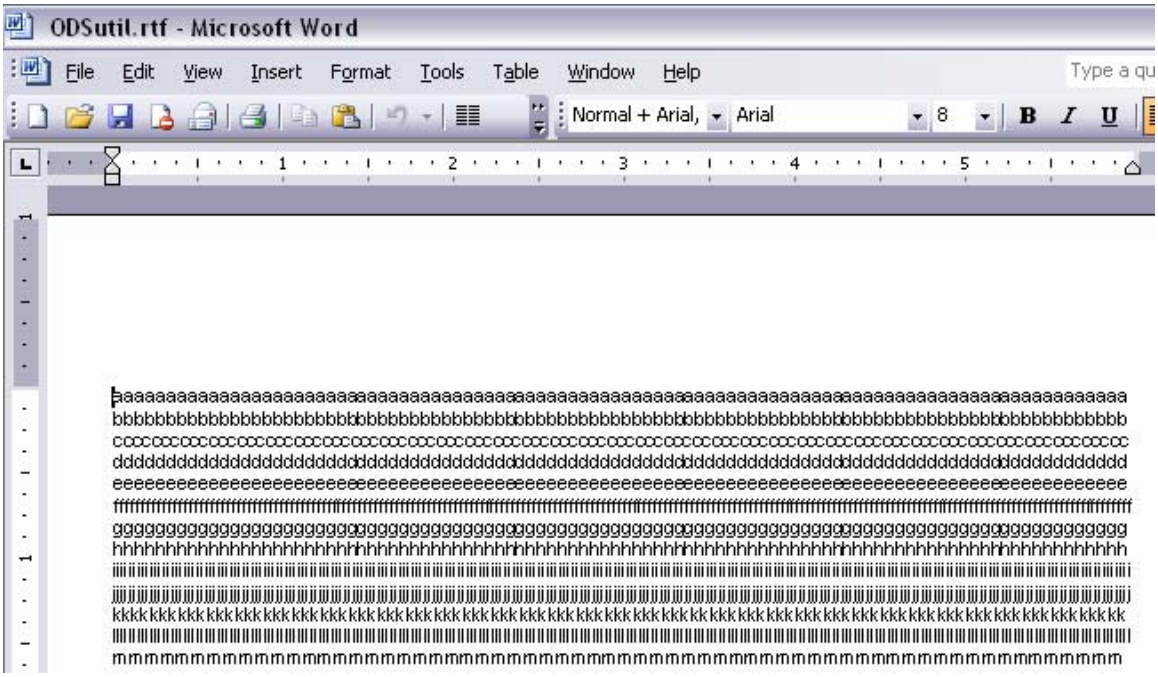

Figure 1

The second step is to save this file as text file, which can be easily read by SAS program. A simple SAS program can read in this text file and find the number of characters on each line. Then the width of a single character is defined as:

Width of a character = Width of text body in Word / Number of characters on one line

In the pharmaceutical industry, all tables or listings generated usually use one standard font for consistency.

Using the same strategy, SAS programmer can create their own utility file to fit their need. If the company only uses one font then this utility file and setup program only has to be run once, and the information that is created can then be saved and can be applied to all future RTF outputs.

2.3 The calculation

The calculation consists of four steps. First, calculate lengths of all characters. Second, calculate number of lines of a string with a fixed cell width. Third, calculate number of lines for an observation. Finally, assign page numbers to all observations.

2.3.1 Calculate lengths of all characters

To calculate the length of each character, the utility txt file and the formula in section 2.2 are used. To implement this in SAS, we create an informat file (Figure 2), which maps each letter to its given length. This will provide us with an easy and understandable way to calculate the length of a string. Once this informat is created and saved permanently, it can be used across studies and across projects.

	fmtname	type	start	label	hlo
1	@inches	I	a	0.0625	
2	@inches	I	b	0.0625	
3	@inches	I	c	0.0560747664	
4	@inches	I	d	0.0625	
5	@inches	I	e	0.0625	
6	@inches	I	f	0.0310880829	
7	@inches	I	g	0.0625	
8	@inches	I	h	0.0625	
9	@inches	I	i	0.03	
10	@inches	I	j	0.03	
11	@inches	I	k	0.0560747664	
12	@inches	I	l	0.03	
13	@inches	I	m	0.09375	
14	@inches	I	n	0.0625	
15	@inches	I	o	0.0625	
16	@inches	I	p	0.0625	
17	@inches	I	q	0.0625	

Figure 2

2.3.2 Calculate number of lines of a string with a fixed cell width

Given a fixed cell width, a string may wrap itself to fit into the cell. In SAS ODS, it breaks between words when wrapping. If a word can not be put on current line with proceeding words because of the limitation of the cell width, the whole word goes to the next line. Because of this, in the wrapped cells, lines may not be fully occupied (Figure 3). This situation will be considered in our calculations. In Figure 3, since the word “between” can not be put on the first line, the whole word goes to the second line. This leaves some blank spaces on the first line.

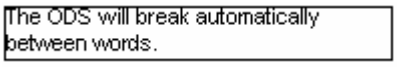

Figure 3

Given a string, the program goes through it one character at a time adding the lengths of each character together. In our implementation, we create two variables to track the string length information needed for the calculation. First is _PB_LENGTH, which records the length of the string for current line. This variable will be compared with the cell width to determine if it exceeds the cell width and a break should be made, thus increasing the number of lines. The second variable is _PB_SUB_LENGTH. It records the sub-length of current word. This variable is set to zero whenever a blank space is met. The purpose of this variable is to keep track of the sub-length of current word, which can not fit into current line and will be put on next line. If a break occurs, the _PB_SUB_LENGTH will be assigned to _PB_LENGTH, which means the whole word is carried over to the next line. The number of lines of a string is increased when the breaks happened, which is as following inequality:

$$\_PB\_LENGTH > \text{Cell width}.$$

2.3.3 Calculate number of lines for an observation

By the calculation in step two, for one observation, there are cells with different number of lines, based on the cell widths and the string lengths. The actual lines of this observation should be the maximum number of lines of all cells (Figure 4).

The SAS System

Cell1	_Cell1_lines	Cell2	_Cell2_lines	Cell3	_Cell3_lines	_max_lines
SAS data sets can have a one-level name or a two-level name.	4	Typically, names of temporary SAS data sets have only one level and are stored in the WORK data library.	6	Fundamental Concepts	2	6

Figure 4

2.3.4 Assign page numbers to all observations

According to the number of lines of each observation and the total number of lines allowed on one page, a page number can be easily assigned to each observation. The data set created after the fourth step will have all the information to set page breaks.

2.4 The SAS Code

See detailed SAS code in appendix.

3 CONCLUSION

In our experiments of using the proposed approach, the results were accurate. In rare cases, when the string length is very close to the cell width, the calculation may go wrong by one line. That is within the range of our expectation. In practice, this deviation is acceptable.

4 REFERENCES

[BS] Brian T. Schellenberger,, A Reintroduction to ODS: The Philosophy of the Output Delivery System; or, How to Find Your Way Around All Those Features. SAS Institute Inc., Cary, NC, USA.

[JK] John Kirkpatrick, Pagination in the ODS. SAS Conference Proceedings: Phuse 2005, Heidelberg, Germany.

5 TRADEMARK INFORMATION

6 CONTACT INFORMATION

Songtao Jiang
Harvard Clinical Research Institute
930 Commonwealth Avenue, 3rd Floor.
Boston, MA 02215
Phone: 617-632-1438
songtao.jiang@hcri.harvard.edu
http://www.hcri.harvard.edu

Daniel Boisvert
Harvard Clinical Research Institute
930 Commonwealth Avenue, 3rd Floor.
Boston, MA 02215
Phone: 617-632-1564
daniel.boisvert@hcri.harvard.edu
http://www.hcri.harvard.edu

Appendix

```
*************************************************************;
* Input the utility file */                                  *;
* ods_util.txt is saved from MS-Word                         *;
*************************************************************;
PROC IMPORT OUT=ods_util
     DATAFILE= "ods_util.txt"
     DBMS=DLM REPLACE;
     DELIMITER='20'x;
     GETNAMES=NO;
     DATAROW=1;
RUN;
*************************************************************;
*   char - The character being described.                   *;
*   charlength - The length in inches of the character.     *;
*   there are 6 inches of characters (in MS-Word)           *;
*   Width of a character = Width of text body in Word       *;
*    divided by Number of characters on one line             *;
*************************************************************;
DATA charlen(KEEP=char charlength);
  SET ods_util;
  char=SUBSTR(var1,1,1);
  charlength=6/LENGTH(var1);
RUN;
*************************************************************;
*  For those characters, which were not included in our     *;
*  utility file, use the MEAN length as approximation       *;
*************************************************************;
PROC SQL NOPRINT;
  SELECT MEAN(charlength)
  INTO : meanlen
  FROM charlen;
QUIT;
*************************************************************;
*  Create an Informat to change each letter to              *;
*  the length in inches.                                    *;
*  In the Informat, create an OTHER line, for all           *;
*  characters, which do not have an assigned length, the    *;
*  MEAN length will be used                                 *;
*************************************************************;
DATA other(DROP=char charlength);
```

```
  SET charlen END=eof;
  fmtname='@inches';
  type='I';
  start=char;
  label=charlength;
  OUTPUT;
  IF eof THEN DO;
   start=' ';
   hlo='';
   label=0.030; /* length of a space */
   OUTPUT;
    start='';
    hlo='O'; /* other characters */
    label=&meanlen.;
    OUTPUT;
  END;
RUN;
*******************************************************;
*  Create the format.                                 *;
*******************************************************;
PROC FORMAT CNTLIN=other;
QUIT;
*******************************************************;
*  Calculate # of lines of a string with cell width   *;
*  _PB_LENGTH - which records the length of the string *;
*  for current line                                   *;
*  _PB_SUB_LENGTH - it records the sub-length of      *;
*  current word. This variable is set to zero whenever *;
*  a blank space is met.                              *;
*******************************************************;
%macro _varLines(inds=,var=,colwidth=);
data &inds.;
    set &inds.;
    format _PB_length _PB_subleng 8.5;
    _PB_length=0;
    _PB_subleng=0;
    _&var._lines=1;
     _PB_strlen=length(&var.);
    do i=1 to _PB_strlen;
        _PB_char=substr(trim(left(&var.)),i,1);
        _PB_inches=INPUT(_PB_char,inches.);
```

```
        _PB_length=_PB_length+_PB_inches;
        if _PB_char=' ' then do;
            _PB_subleng=0;
        end;
        else do;
            _PB_subleng=_PB_subleng+_PB_inches;
        end;
        if _PB_length>&colwidth. then do;
            _&var._lines=_&var._lines+1;
            /* if a word cannot fit within the column */
            if _PB_subleng=_PB_length then _PB_subleng=0;
            _PB_length=_PB_subleng;
        end;
    end;
    drop _PB_length _PB_subleng _PB_strlen
        _PB_char i _PB_inches;
run;
%mend _varLines;
*********************************************************;
*  Calculate number of lines for an observation        *;
*********************************************************;
%macro _obsLines(inds=,vars=,colwidths=);
data _null_;
    array strs &vars.;
    num=dim(strs);
    call symput('numVars', num);
run;
data _null_;
%do i=1 %to &numVars.;
    %let _PB_v=%scan(&vars,&i,' ');
    %let _PB_c=%scan(&colwidths,&i,' ');
    /* each variable */
    %_varLines(inds=&inds.,var=&_PB_v.,colwidth=&_PB_c.);
%end;
run;
data &inds.;
    set &inds.;
    _max_lines=0;
    %do i=1 %to &numVars.;
        %let _PB_v=%scan(&vars,&i,' ');
        if _&_PB_v._lines>_max_lines then do;
```

```
          _max_lines=_&_PB_v._lines;
       end;
   %end;
run;
%mend _obsLines;
*********************************************************;
*  Assign page numbers to all observations              *;
*  The default page size is 50 lines                    *;
*********************************************************;
%macro _pageBreak(   inds=,
                     vars=,
                     colwidths=,
                     pagesize=50,
                 );
%_obsLines(inds=&inds.,vars=&vars.,colwidths=&colwidths.);
data &inds.;
   set &inds.;
   retain _PB_seq 0 _PB_page 1;
   if _PB_seq+_max_lines>&pagesize. then do;
      _PB_seq=_max_lines;
      _PB_page=_PB_page+1;
   end;
   else _PB_seq=_PB_seq+_max_lines;
   drop _PB_seq;
run;
%mend _pageBreak;
/* Example */
%_pageBreak(inds=test0,
          vars=Cell1 Cell2 Cell3,
          colwidths=1 1 1,
          pagesize=40);
```