Text Wrapping with Indentation for RTF Reports
Abhinav Srivastva, Gilead Sciences Inc., Foster City, CA

ABSTRACT
It is often desired to display long text in a report field in a way to avoid splitting in an unusual manner when rendered with reporting procedures in SAS®. Sometimes, the field can be a combination of several other fields or variables, in which case, the resultant text field needs to be well formatted for readability to the consumer. The paper presents a brief overview of common text wrapping and indentation options for RTF outputs and will incorporate one of the methods in building a SAS® macro capable of text wrapping and indentation on one or more text fields.

INTRODUCTION
For indentation and wrapping in RTF, several options exist and below Table is a summary of some of them:

<table>
<thead>
<tr>
<th>Indentation Methods</th>
<th>Usage</th>
</tr>
</thead>
</table>
| RTF Control Words: \li and \fi | String = ‘Some long text to be formatted’
  • With \li250: string = ‘Some long text to be formatted’
  • With \fi-250\li250: string = ‘Some long text to be formatted’
  Combination of ‘\li’ and ‘\fi’ can produce hanging text as above.
  (\li=left indent, \fi=first line indent) |
| Style overrides: indent = xx, leftmargin=xx | style = [indent = xx]
  style = [leftmargin=xx] |
| Pretext in PROC REPORT | Pretext = [ {empty string or RTF control words} ] |
| Styling with ASIS=ON | string = ‘Some long text to be formatted’
  Spaces are added beforehand in the data so that ASIS=ON in PROC REPORT will honor it. |

Table 1: Indentation methods for RTF
For splitting long text across lines, there are several options too; either explicitly specifying certain characters in the data such as ^n, ^{newline 1}, RTF control words like ‘line’ (where ^= escape character) or through styling column width as <cellwidth=xx>.

While all of these would work fine when dealing with a single text field but when multiple fields or columns are combined and reported as a single text field, it could become quite challenging in enabling readability to the end user. With that in perspective, the "%text_wrap" macro (discussed below) handles them smoothly using a white space character “w” and a new line character “n”, eg. “^w” and “^n”, and formats them as desired.

TEXT WRAPPING EXAMPLES
Let’s assume there is a “Variable A” in the dataset which has long text and needs to be reported with a set column width of 20% in the report.

The following code will restrict to 20% as desired:

display Variable_A / style(column)=[cellwidth=20%];

<table>
<thead>
<tr>
<th>Data (Variable A)</th>
<th>Report Column (Variable A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA BBBB CCCCCC DDDDDD EEEEEE FFFFFFFFFF</td>
<td>AAAA BBBB CCCCCC DDDDDD EEEEEE FFFFFFFFFF</td>
</tr>
</tbody>
</table>

Table 2: REPORT Procedure <cellwidth=> style option
Text Wrapping with Indentation for RTF Reports, continued

The output can be enhanced with some indentation when text flows over to the next line or every subsequent line thereafter with RTF control words ‘fi-xx\lixx’ where xx=# of twips (1 twip= 1/20th of a point, or 1440 twips= 1 inch).

\[\text{display Variable\_A / style(column)=[cellwidth=20\% protectspecialchars=off pretext='\pnhang\fi-250\li250 \'};\]

<table>
<thead>
<tr>
<th>Data (Variable A)</th>
<th>Report Column (Variable A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA BBBB CCCCCC DDDDDD EEEEEE FFFFFFFFFF</td>
<td>AAAA BBBB CCCCCC DDDDDD EEEEEE FFFFFFFFFF</td>
</tr>
</tbody>
</table>

Table 3: REPORT Procedure with RTF control words

Next, let’s consider a case when multiple text columns are combined into one (using ‘^n’ or new line character) to be reported as a single report column as demonstrated below:

\[
data B;
\text{set A;}
\text{New\_Var = catx('^n', Variable\_A, Variable\_B, Variable\_C);}
\text{run;}
\]
\[\text{ods escapechar='^';}
\]
\[\text{...}
\]
\[\text{display New\_Var / style(column)=[cellwidth=20\%]};
\]

<table>
<thead>
<tr>
<th>Data (Variable A)</th>
<th>Data (Variable B)</th>
<th>Data (Variable C)</th>
<th>Report Column (New Var)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA BBBB CCCC</td>
<td>PPPPPP PPPPPP QQQQ RRRRR</td>
<td>XXXXX YYYY YYYY ZZZZZZZZZZ</td>
<td>AAAA BBBB CCCC PPPPPP PPPPPP QQQQ RRRRR XXXXX YYYY YYYY ZZZZZZZZZZ</td>
</tr>
</tbody>
</table>

Table 4: Combining multiple columns into one

An immediate improvement, just like the case with single column (Table 3), can be made in the resultant report column is with the indentation of text for each variable so as to clearly distinguish them in the report. For new line and indentation ‘^n^w^w’ string as delimiter in CATX function is used below to add two leading spaces for each variable.

\[
data B;
\text{set A;}
\text{New\_Var = catx('^n^w^w', Variable\_A, Variable\_B, Variable\_C);}
\text{run;}
\]

<table>
<thead>
<tr>
<th>Data (Variable A)</th>
<th>Data (Variable B)</th>
<th>Data (Variable C)</th>
<th>Report Column (New Var)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA BBBB CCCC</td>
<td>PPPPPP PPPPPP QQQQ RRRRR</td>
<td>XXXXX YYYY YYYY ZZZZZZZZZZ</td>
<td>AAAA BBBB CCCC PPPPPP PPPPPP QQQQ RRRRR XXXXX YYYY YYYY ZZZZZZZZZZ</td>
</tr>
</tbody>
</table>

Table 5: Combining multiple columns into one with indentation
As observed in the resultant report column (New_Var) from the above display (Table 5), Variables B and C begin with two leading spaces as expected, but the text wrapping isn’t quite accurate as the text always starts at the first position in the column when it does not fit completely as per the pre-specified column width. So the text ‘ZZZZZZZZ’ coming from Variable C will start at the first position in the column making it not appealing to the user.

The macro discussed below handles this situation and others that could potentially arise in the data.

**TEXT WRAP MACRO**

Basic usage of the “Text Wrap” macro is as below:

```sas
%macro text_wrap (dsn      = ,   /* Name of the Input dataset */
                  invars   = ,   /* Input Variable(s), separate with space,
                    example: VarA VarB VarC */
                  outvar   = ,   /* Name of the resultant variable in the report */
                  width    = 30, /* Desired # characters to fit per line in outvar,
                    default=30 */
                  ind      = 1,  /* Indent spaces to add, default=1 */
                  esc      = ^,  /* Escape character, default=^ */
                  countvar = c   /* Variable to display total # lines consumed by
                    outvar, useful when deciding # lines per page */
                );
```

The resultant column from Table 5 can be formatted with `%text_wrap` macro to improvise on wrapping and indentation. In the resultant column, the macro will indent every variable in the same proportion, i.e. if `<ind=` parameter in the macro call is ‘2’ then Variable B text will be indented with 2 spaces relative to Variable A; Variable C text will be indented with 2 spaces relative to Variable B for a total of 4 indents from the first column position (or Variable A). Also, the text from each variable will wrap corresponding to its starting position and not flow from the beginning of the column as happened in Table 5.

<table>
<thead>
<tr>
<th>Data (Variable A)</th>
<th>Data (Variable B)</th>
<th>Data (Variable C)</th>
<th>Report Column (New Var)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA B BBBB CCCC</td>
<td>PPPP PPP PQQQQ</td>
<td>XXXX YYYYY YYYY</td>
<td>AAAA B BBBB CCCC</td>
</tr>
<tr>
<td></td>
<td>RRRRR</td>
<td>ZZZZZZZZZ</td>
<td></td>
</tr>
</tbody>
</table>

**Table 6: Revised result with Text Wrap Macro**

Sometimes, the data can have missing values for one or more variables; in that case `%text_wrap` macro will hold # spaces as a way to indicate a missing value. Consider the case when Variable B has a missing value, then with `<indent=2>` parameter the resultant column will have text from Variable C indented 4 spaces assuming 2 indents for Variable B text which happens to be missing, and this indent pattern should indicate the user of a missing value in the data for Variable B.

<table>
<thead>
<tr>
<th>Data (Variable A)</th>
<th>Data (Variable B)</th>
<th>Data (Variable C)</th>
<th>Report Column (New Var)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA B BBBB CCCC</td>
<td>XXXX YYYYY YYYY</td>
<td>AAAA B BBBB CCCC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ZZZZZZZZZ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 7: Text Wrap Macro result with missing data**

Please refer to Appendix for complete SAS code of the “Text Wrap” Macro.
CONCLUSION

SAS provide a number of ways to format text and this paper leverages on one of them to build a macro capable of dealing with RTF destination. The summary table (Table 1) should be a good starting point to explore different methods. Also, the readers are encouraged to try different methods against different ODS destinations to see the interaction and find a solution for their individual needs.

REFERENCES


ACKNOWLEDGMENTS

I would like to thank Ms. Asha Srivastva for the encouragement and review of this paper.

CONTACT INFORMATION

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

Name: Abhinav Srivastva
Enterprise: Gilead Science Inc
Address: 333 Lakeside Dr.
City, State ZIP: Foster City, CA 94404
Work Phone: (650) 389-8778
E-mail: Abhinav.Srivastva@gilead.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.
APPENDIX

%macro text_wrap (dsn = ,
invars = ,
outvar = newvar,
width = 30,
ind = 1,
esc = ^,
countvar = c );
%global levels;
/* Basic parameters check */
%if &dsn. = ' ' %then %do;
  %put No Dataset provided ;
  %goto exit;
%end;
%if &invars. = ' ' %then %do;
  %put No Input Variables Provided ;
  %goto exit;
%end;
%if &ind. < 1 %then %do;
  %put Indent parameter has to be > 1;
  %goto exit;
%end;
/* Check the existence of variables in the given dataset */
%let dsid = %sysfunc(open(&dsn.));
%if (&dsid) %then %do;
  %let k=1;
  %let comp_str = &invars;
  %do %while(%scan(&comp_str.,&k.,%str( )) ne %str( ));
    %let var2comp = %scan(&comp_str.,&k.,%str( ));
    %if %sysfunc(varnum(&dsid.,&var2comp.)) %then %put Valid Variable(s) entered;
    %else %do;
      %put !!! Variable &var2comp. not found. !!!;
      %goto exit;
    %end;
    %let k = %eval(&k.+1);
  %end;
%end;
%let rc = %sysfunc(close(&dsid));
%let levels=%eval(&k.-1);
/* Process each variable and format as needed (wrap + Indent) */;
data &dsn. (drop = len nwords count word __&outvar.: __c:);
  set &dsn. ;
%do i=1 %to &levels ;
%let var&i.= %scan(&invars.,&i.,%str( ));
%let indent&i.= %eval(&i. * &ind. - &ind.);

length __&outvar&i.__ $ 1000 &outvar $ 5000 word $200; /* please adjust if needed;
__&countvar&i.__ = 1; __&outvar&i.__ = '';
len=0; count=0; word='';
nwords = countw(&&var&i.,' ');

if nwords>0 then do;
  * count loop;
  do count = 1 to nwords;
    word = scan(&&var&i., count, '');
    if len + 1 + length(word) > &width. then do;
      __&countvar&i.__ += 1;
      * append word and if length > width then insert a new line *;
      %if &i.=1 %then %do;
        %if &i. = &levels. %then %do;
          __&outvar&i.__ = strip(__&outvar&i.__)||"&esc.n"||
          repeat(%str("&esc.w"),%eval(&ind.-1))|| word;
        %end;
        %else %do;
          __&outvar&i.__ = strip(__&outvar&i.__)||"&esc.n"|| word;
        %end;
      %end;
      len = length(word);
    end;
  else do;
    __&countvar&i.__ = 0;
  end;
  %if &i.=1 %then %do;
    __&outvar&i.__ = left(__&outvar&i.__);
  %end;
  %else %do;
    if __&outvar&i.__ ^= ' ' then
      __&outvar&i.__ = repeat(%str("&esc.w"),%eval(&&indent&i.-1))||left(__&outvar&i.__);
  %end;
end;

&outvar = catx("&esc.n",of __&outvar.:);
&countvar = sum(of __&countvar.:);

run;
%exit:
%mend;