Paper CC06

Empty Lines in SAS Tables (ODS LISTING and ODS RTF Output) produced by Proc Tabulate

Twan van Berkel, NV Organon, Oss, The Netherlands

ABSTRACT

Within our pharmaceutical company NV Organon clinical studies are performed and statistical appendices are created using the SAS System® (1). These appendices contain among other things large tables. To reduce the size of these tables and to make the tables more readable, empty lines instead of normal line separators were introduced.

The empty line tables are produced by specially written SAS macros containing large amounts of SAS code.

For some tabular designs the creation of empty lines can be done a lot easier by using PROC TABULATE instead of SAS macros.

Four different methods describe how to introduce empty lines formatted as ODS LISTING output. In addition, two methods are introduced that explain how to create empty lines using PROC TABULATE and ODS RTF, formatted as RTF output. Every method has its own features and the user can use the different methods for his/her own tabular design.

Empty lines do not always enhance the ODS RTF output considering that normal horizontal line separators are included without the ability to switch them off.

If you want to introduce empty lines in an RTF formatted document, introduce them within the PROC TABULATE syntax instead of introducing them (solely) from the ODS RTF facility.

INTRODUCTION

Is it possible to produce horizontal empty lines with PROC TABULATE formatted as ODS LISTING output? This question came to my mind when I encountered tables containing empty lines, like Table C shown in Figure 1. These tables are commonly used within our department as part of a clinical report.

Using empty lines instead of normal lines as separators has a quantitative and a cosmetic reason. Let's take a look at Figure 1 Table A, B and C.

Table A has normal line separators, Table B has no line separators and Table C has empty line separators. Table A decreases in size approximately 30% when normal lines are removed (Table B). The disadvantage is a less readable Table B. So, to minimize the table size and to maximize the readability of the table, empty lines are introduced in Table C. Compared to Table A, Table C decreases in size by approximately 20%. Our statistical appendices can contain large tables, so a size decreasement is very welcome.

The tables with empty lines are produced by specially written SAS macros containing large amounts of SAS code.

For some tabular design the creation of empty lines can be done a lot easier by using PROC TABULATE instead of SAS macros.

This paper does not give an answer to all tabulation designs developed with PROC TABULATE. You could use the examples as handle for your own table. Some experience with PROC TABULATE and ODS RTF is recommended for a better understanding of this paper.

This paper describes four different methods for creating empty lines with PROC TABULATE formatted as ODS LISTING output. In addition, two methods are introduced how to create empty lines using PROC TABULATE and ODS RTF.

SAS LISTING output and RTF document format are the main sources for our clinical reports and the accompanying statistical appendices.

Figure 1: The tables A, B and C.

Table A: Normal line separators

Table A		Treatment Group					
		Group A		Group B		Group C	
		n	%	n	%	n	%
Female	Young	3	75.0	1	20.0	2	66.7
	Middle	1	25.0	1	20.0	1	.3
	Older			3	60.0		
Male	Young	1	50.0	1	50.0	2	100.0
	Middle			1	50.0		
	Older	1	50.0				
All	Young	4	66.7	2	28.6	4	80.0
	Middle	1	16.7	2	28.6	1	20.0
	Older	1	16.7	3	42.9		

Table B: No line separators

Table B		Treatment Group					
		Group A		Group B		Group C	
		n	%	n	%	n	%
Female	Young	3	75.0	1	20.0	2	66.7
	Middle	1	25.0	1	20.0	1	33.3
	Older			3	60.0		
Male	Young	1	50.0	1	50.0	2	100.0
	Middle			1	50.0		
	Older	1	50.0				
All	Young	4	66.7	2	28.6	4	80.0
	Middle	1	16.7	2	28.6	1	20.0
	Older	1	16.7	3	42.9		

Table C: Empty line separators

Table C		Treatment Group					
		Group A		Group B		Group C	
		n	%	n	%	n	%
Female	Young	3	75.0	1	20.0	2	66.7
	Middle	1	25.0	1	20.0	1	33.3
	Older			3	60.0		
Male	Young	1	50.0	1	50.0	2	100.0
	Middle			1	50.0		
	Older	1	50.0				
All	Young	4	66.7	2	28.6	4	80.0
	Middle	1	16.7	2	28.6	1	20.0
	Older	1	16.7	3	42.9		

EXAMPLE DESIGN

All the methods, except the fourth one, use a frequency table as format where the statistics are presented in the column dimension. The fourth method uses a “summary statistics table” where the statistics are presented in the row dimension. The choice of table formatting in the examples is used to emphasize the six different empty line methods as much as possible.

First a discussion per method is held considering ODS LISTING output. The next step is to extend the syntax of the first four methods with ODS RTF in order to see the effect of the empty lines in an RTF formatted document.

Method 5 is a hybrid method that demonstrates how to implement empty lines using the PROC TABULATE and the ODS RTF facility. Furthermore, method 6 demonstrates how empty lines can be implemented using the ODS RTF facility.

The examples use the data and variable properties and formats displayed in Figure 2. The added syntax that creates empty lines uses a bold font face in the illustrating figures.

There are three numerical variables (TCPF, AGE and AGECLASS) and one character variable GENDER in the used dataset.

CONFIGURATION

The following options are minimally set within the SAS system:

```
OPTIONS MISSING=' ' FORMCHAR='|_ _ ||||||+=|-/\<>*';
```

The configuration OPTIONS MISSING=’ ’ shows a white space instead of a period for every missing value. The FORMCHAR option defines the characters to be used for constructing the table outlines and dividers (2).

Testing was done in a Windows environment (Microsoft Windows XP® SP1.0) with SAS/BASE® version 8.2 and 9.1.3.

Figure 2: Properties and printed output of the dataset used by the methods.

```
List output:

Obs     TCPF      GENDER    AGE    AGECLASS

  1    Group A    Female     25     Young
  2    Group B    Female     74     Older
  3    Group C    Male       19     Young
  4    Group C    Female     19     Young
  5    Group B    Female     35     Middle
  6    Group A    Female     18     Young
  7    Group A    Female     41     Middle
  8    Group B    Male       41     Middle
  9    Group C    Male       25     Young
 10    Group A    Male       22     Young
 11    Group B    Female     80     Older
 12    Group C    Female     51     Middle
 13    Group B    Male       13     Young
 14    Group B    Female     15     Young
 15    Group A    Female     14     Young
 16    Group C    Female     15     Young
 17    Group B    Female     80     Older
 18    Group A    Male       60     Older


Variable properties:

  Variable     Type    Length   Format
    TCPF        Num       8      TCPF.
    GENDER      Char      8      $SEX.
    AGE         Num       8
    AGECLASS    Num       8      AGECL.


PROC FORMAT;
 value tcpf 1 = 'Group A'
            2 = 'Group B'
            3 = 'Group C';

 value agecl 1 = 'Young'
             2 = 'Middle'
             3 = 'Older';

 value $sex 'F' = 'Female'
            'M' = 'Male';
run;
```

METHOD 1: EMPTY LINES CREATED WITH A DUMMY ANALYSIS VARIABLE AND THE EMPTY FORMAT

First, a table output like Table B shown in Figure 1 is created. The syntax for this table is shown in Figure 3. The example is a frequency table with percentages. The treatment group (TCPF) variable is placed in the column dimension. In the row dimension the class variables AGECLASS per GENDER are presented.

Totals are presented at the bottom of the table for the GENDER variable.

PCTN has AGECLASS as the denominator because the percentages per TCPF and AGECLASS are presented. The normal line separators are removed by using the NOSEPS option within PROC TABULATE.

CREATING THE EMPTY LINE

The syntax for creating an empty line is shown in Figure 4. The created table is shown in Figure 1, named Table C. First, add a dummy variable to create the actual empty line. DUMMY is set to missing for every observation (DUMMY=.). DUMMY is defined as analysis variable otherwise no table is produced.

Figure 3: The original PROC TABULATE syntax.

```
PROC TABULATE data=a noseps;
  class gender ageclass tcpf;
  format gender $sex.
         ageclass agecl.
         tcpf tcpf.;
  table (gender=' ' all='All') * (ageclass='
'),
        (tcpf='Treatment Group')*
        (n='n'*f=5.0 pctn<ageclass>='%'*f=5.1)
  /rts=30 box='Table B';
run;
```

Figure 4: Method 1 syntax.

```
proc format;
 value empty 0 = ' ';
run;

data b;
 set a;
 dummy=.;
run;

PROC TABULATE data=b noseps;
  class gender ageclass tcpf;
  var dummy;
  format gender $sex.
         ageclass agecl.
         tcpf tcpf.;
  table (gender=' ' all='All') * (ageclass=' ' dummy=' '),
        (tcpf='Treatment Group')*
        (n='n'*f=empty5.0 pctn<ageclass dummy>='%'*f=5.1)
  /rts=30 box='Table C';
run;
```

Where is the DUMMY situated in the table?
The empty line is the DUMMY variable. In Figure 5 the label of the DUMMY is ‘Dummy’ and the N values for the DUMMY variable are zeros. To remove these zeros the ‘EMPTY’ format shown in Figure 4 is introduced.
Normally the DUMMY value for PCTN would be a period (missing value). Instead of the period a white space is presented because of the OPTIONS MISSING=’ ’ configuration (this can also be done within PROC TABULATE by using the MISSTEXT=’ ’ option). A disadvantage of this method is that all the missing values (periods) in the table (including the row table space) are removed by the MISSING option.

The empty line appears after the last class variable in the row dimension. To get a consistent crossing the DUMMY variable is placed after the AGECLASS variable in the PCTN denominator.
To understand a crossing one can imagine the element*element in the table statement dimension as a stem of a tree which branches to the leaves. The stem is the first (class or analysis) variable encountered and the leaves of the tree are the cells of the table.

Figure 5: Location of the DUMMY variable.

Table C		Treatment Group					
		Group A		Group B		Group C	
		n	%	n	%	n	%
Female	Young	3	75.0	1	20.0	2	66.7
	Middle	1	25.0	1	20.0	1	33.3
	Older			3	60.0		
	Dummy	0		0		0	
Male	Young	1	50.0	1	50.0	2	100.0
	Middle			1	50.0		
	Older	1	50.0				
	Dummy	0		0		0	
All	Young	4	66.7	2	28.6	4	80.0
	Middle	1	16.7	2	28.6	1	20.0
	Older	1	16.7	3	42.9		
	Dummy	0		0		0	

When a class variable has missing values in the dataset one can use the PROC TABULATE option MISSING to include the missing values of (a) class variable(s) in the PROC TABULATE calculations.
The option MISSING in PROC TABULATE does not affect the empty lines created in the table.
Tip: Define a format for a missing value to distinguish it clearly in the table, otherwise the missing value is labelled as white space in the row table space due to the configuration OPTIONS MISSING=' '.

METHOD 2: EMPTY LINES CREATED WITH A DUMMY ANALYSIS VARIABLE USING THE SUM STATISTIC

There is an alternative when one does not like to use PROC FORMAT statement in Figure 4 to wipe out the zeros in a table with empty lines. The alternative syntax is shown in Figure 6 and produces almost the same Table C in Figure 1. The table produced with syntax from Figure 6 differs in the position of the class variable AGECLASS in the row titles. The row table space has to be shared with an extra variable VALUE (See for yourself and compare the different table outputs!).

Figure 6: Method 2 syntax.

```
data b;
 set a;
 dummy=.;
 value=1;
run;

PROC TABULATE data=b noseps;
  class gender ageclass tcpf;
  var dummy value;
  format gender $sex.
         ageclass agecl.
         tcpf tcpf.;
  table (gender=' ' all='All') * (ageclass=' '*value=' ' dummy=' '),
        (tcpf='Treatment Group')*
        (sum='n'*f=5.0 pctn<ageclass dummy>='%'*f=5.1)
  /rts=30 box='Table C';
run;
```

CREATING THE EMPTY LINE

Replace the N statistic with the SUM statistic and introduce the variable VALUE.
The SUM statistic requires an analysis variable in the last crossing (the N statistic does not require this). The DUMMY variable shows missing values instead of 'wiped out zeros'. These missing values are represented as a white space (OPTIONS MISSING=' '). Finally, the EMPTY format can be removed. Keep in mind that the value of the variable

VALUE has to be 1 for every observation in the dataset.

METHOD 3: EMPTY LINES CREATED WITH PRELOADFMT AND PRINTMISS

Table C in Figure 1 can also be created using the PRELOADFMT option in combination with the PRINTMISS option. In contrast with Method 1 and Method 2, this method does not require help variables like DUMMY and/or VALUE. See Figure 7 for the syntax.

Figure 7: Method 3 syntax.

```
PROC FORMAT;
 value agecl 1 = 'Young'
             2 = 'Middle'
             3 = 'Older'
             4 = ' ';
run;

PROC TABULATE data=a noseps;
 class gender tcpf;
 class ageclass / preloadfmt;
 format gender $sex.
        ageclass agecl.
        tcpf tcpf.;
  table (gender=' ' All='All' ) * (ageclass=' '),
        (tcpf='Treatment Group')*
        (n='n'*f=5.0 pctn<ageclass>='%'*f=empty5.1)
  /rts=30 box='Table C' printmiss;
run;
```

CREATING THE EMPTY LINE

The PRELOADFMT is a class statement option that “pre-loads” user-defined formats in the row and column area (2).
The PRELOADFMT option is added to the AGECLASS class statement because SAS has to know which format to “pre-load” (see Figure 7).
The PRINTMISS option forces to display all the AGECLASS formatted values in the table. So, define white space as the last category in the AGECL format (4 =’ ‘).

How do the PRELOADFMT and PRINTMISS options react to the missing values?
Introducing missing values in the dataset for the AGECLASS variable does not affect the empty line. Thus, to emphasize a missing value, the format can be adapted:

```
value agecl
  . = 'Missing’
  1 = 'Young'
  2 = 'Middle'
  3 = 'Older'
  4 = ' '; <The actual empty line>
```

There is a disadvantage: For every crossing this format “pre-loads” the formatted value ‘Missing’, just like the empty line.

A close reader will detect a difference where the EMTPY format is placed in the statistic syntax of Method 1 and Method 3:

```
(n='n'*f=empty5.0 pctn<ageclass dummy>='%'*f=     5.1)  <Figure 4>
(n='n'*f=     5.0 pctn<ageclass>      ='%'*f=empty5.1)  <Figure 7>
```

It appeared that Method 1 produces a zero for the N statistic and not for the PCTN statistic, while Method 3 does this vice versa (if you leave out the EMTY format). It is not clear why. The zeros were removed with the EMPTY format.

METHOD 4: EMPTY LINES CREATED WITH THE INDENT OPTION IN COMBINATION WITH A DUMMY VARIABLE CONTAINING A SPECIAL BLANK

Let us now look at the table shown in Figure 8.
The variables GENDER and AGECLASS are crossed with N, MEAN, MIN and MAX statistics in the row dimension and are divided by an empty line.

Figure 8: Empty lines created with a dummy variable containing a special blank and using the INDENT option.

	Treatment Group		
	Group A	Group B	Group C
Gender			
Female			
- N	4	5	3
- Mean	24.5	56.8	28.3
- Minimum	14.0	15.0	15.0
- Maximum	41.0	80.0	51.0
Male			
- N	2	2	2
- Mean	41.0	27.0	22.0
- Minimum	22.0	13.0	19.0
- Maximum	60.0	41.0	25.0
Ageclass			
Young			
- N	4	2	4
- Mean	19.8	14.0	19.5
- Minimum	14.0	13.0	15.0
- Maximum	25.0	15.0	25.0
Middle			
- N	1	2	1
- Mean	41.0	38.0	51.0
- Minimum	41.0	35.0	51.0
- Maximum	41.0	41.0	51.0
Older			
- N	1	3	
- Mean	60.0	78.0	
- Minimum	60.0	74.0	
- Maximum	60.0	80.0	

This example uses the properties of the INDENT option.
The INDENT option allows one to specify a number of spaces to indent each level of row crossing (3). Each level of row crossing is shifted down and is indented by the specified number of spaces. In the syntax, shown in Figure 9, three dummy variables are introduced for every observation.
The INDENT option removes class variable labels in the row titles. That is why the variables GENDERH and AGECLASH are added and contain the labels as string of the class variables presented. These variables have to be defined as CLASS variables otherwise no table is produced.

CREATING THE EMPTY LINE

A special byte value is used to create the actual empty line. No empty line would have been produced if the syntax EMPTLINE=BYTE (0) was replaced by the syntax EMPTLINE=' ' in the data step shown in Figure 9.

The simple rule is:
1) EMPTLINE * new (class) variable containing the label of the CLASS variable * CLASS variable
2) INDENT=0 (zero number of spaces to indent).

Be aware when using other special symbols within the SAS system for the replacement of a blank. Special characters are outputted to the printer and different print languages could interpret these symbols in a different way.
No empty lines can be introduced in the different categories within a class variable.

What if a class variable is missing? Do I still get a correct table?
A correct table is produced when missing class values are introduced in the row and column dimension.

Figure 9: Method 4 syntax.

```
data b;
 set a;
 emptline=byte(0);
 ageclash='Age';
 genderh='Gender';
run;

PROC TABULATE data=b noseps format=7.1;
 var age;
 class gender genderh tcpf emptline ageclass ageclash;
 format gender $sex.
        ageclass agecl.
        tcpf tcpf.;
  table (genderh*gender emptline*ageclash*ageclass)*
        (n='-  N'*f=7.0
         mean='-  Mean'
         min='-  Minimum'
         max='-  Maximum'),
        (tcpf='Treatment Group')*age=''
  /rts=30 box=' ' indent=0;
run;
```

PASSING THE METHOD 1, METHOD 2, METHOD 3 AND METHOD 4 THROUGH THE ODS RTF FACILITY

Since version 7 SAS can deliver procedure and dataset output in a variety of common formats like HTML, XML, PDF and RTF by means of the Output Delivery System (ODS). SAS can output in RTF format by using ODS RTF syntax. RTF, abbreviation for Rich Text Format, is a vendor neutral, platform independent mark-up language that easy assimilates with a Microsoft Word® document. Only two global ODS RTF statements are necessary to do the job:

```
ods rtf file='exam1.rtf';
<PROC TABULATE syntax>
ods rtf close;
```

An RTF document named exam1.rtf is produced. The ODS RTF statements above were used to extend the syntax of Method 1, …, Method 4.

How do the empty lines behave in RTF formatted files using the syntax from Method 1, …, Method 4?
The empty lines produced by Method 1 and Method 3 gave an RTF table that is shown in Figure 10. It produces neat empty lines.
Method 2 and Method 4 gave wrong formatted tables which are successively shown in Appendix 1 and 2.

What can we conclude from this "passing through the ODS RTF facility"?
Not much, except that neat formatted empty line tables are created by trial and error.
However, a small trend can be detected looking at the syntax of Methods 1, 2 and 3: It seems that for every crossing in the row dimension a (virtual) variable value occupies a table cell and this must be consistent over the columns. Compare the crossings in Method 1 and Method 2 in the row dimension.

Figure 10: RTF output of methods 1 and 3.

Table C		Treatment Group					
		Group A		Group B		Group C	
		n	%	n	%	n	%
Female	Young	3	75.0	1	20.0	2	66.7
	Middle	1	25.0	1	20.0	1	33.3
	Older			3	60.0		
Male	Young	1	50.0	1	50.0	2	100.0
	Middle			1	50.0		
	Older	1	50.0				
All	Young	4	66.7	2	28.6	4	80.0
	Middle	1	16.7	2	28.6	1	20.0
	Older	1	16.7	3	42.9		

ODS RTF: THE ADDED VALUE OF AN EMPTY LINE

Why using empty lines in the RTF table when the NOSEPS option does not have any effect using ODS RTF?
The empty line is not really an added value considering the table shown in Figure 9. The empty line in RTF can have an added value in combination with changes made to the layout by using certain user defined styles.
Within our clinical reports Microsoft Word tables often do not have horizontal line separators. In ODS LISTING output horizontal line separators are removed by using the NOSEPS option within PROC TABULATE. For RTF formatted tables, the NOSEPS feature is mimicked by changing raw RTF code. So, empty lines instead of normal line separators are necessary.

In the four methods the empty lines were added from the basic SAS facility (using PROC TABULATE and/or a data step and/or a format).
What if the empty lines are introduced solely by the ODS RTF facility?
This implies that we do not have to trick PROC TABULATE. Thus, during testing it was very hard to come up with solutions without using PROC TABULATE tricks (being restricted to the tabular designs used in the methods). I came up with two methods defined as Method 5 and 6 to implement empty lines (partly) by the ODS RTF facility.

METHOD 5: FOREGROUND COLOR THE SAME AS BACKGROUND COLOR BY USING A STYLE ATTRIBUTES

This method is partly done in the PROC TABULATE syntax and partly with ODS RTF using one style option. The trick is to colour the foreground like the labels and figures in a table the same as the background.
So, if the background is white, make the foreground white and can be done using style attributes. A style attribute is the smallest item within an ODS style definition and has a paired syntax:

```
<name style attribute> = <value style attribute>
```

for example: `BACKGROUND = WHITE.`

PROC TABULATE allows you to control styles attributes within the procedure via the STYLE=option. The STYLE= option affects different areas in the table and the syntax can be added to different areas within the PROC TABULATE syntax(4). Consider the syntax shown in Figure 11.

Figure 11: Method 5 syntax.

```
data b;
 set a;
 dummy=.;
run;

ods rtf file='EM5.rtf';

PROC TABULATE data=b;
 class gender ageclass tcpf;
 var dummy;
 format gender $sex.
 ageclass agecl.
 tcpf tcpf.;
 table (gender=' ' all='All') *
       (ageclass=' ' dummy=' ' * [style=[foreground=white]]),
       (tcpf='Treatment Group')*
       (n='n'*f=5.0 pctn<ageclass dummy>='%'*f=5.1)
       /rts=30 box='Table C';
run;

ods rtf close;
```

CREATING THE EMPTY LINE

Take a look at the syntax shown in Figure 11. The syntax * [STYLE=[FOREGROUND=WHITE]] colours the DUMMY variable values white.

The method works, but it is not an obvious method to use. An editor can change the tabulation text and figures into for example blue within an application like Microsoft Word manually. The label and the figures of the empty line become visible. This in contrast when you are doing all your empty line formatting in PROC TABULATE.

METHOD 6: ADDING AN RTF CONTROL WORD THAT PRODUCES AN EMPTY LINE

Since SAS version 8.2 ODS RTF has the (limited) ability to add raw RTF code to the procedure output. A raw isolated RTF code is called a control word and instructs applications like Microsoft Word to manage documents. The RTF control word for an empty line is “\line”. This instruction gives a linefeed to an RTF document. Within ODS RTF one has to define an escape character that allows addition of RTF control words. This can be defined with the global ODS ESCAPECHAR statement.

CREATING THE EMPTY LINE

The backslash within ODS ESCAPECHAR=’\’ syntax shown in Figure 12 is the escape symbol to add RTF control words from SAS directly into the RTF document. In this example the original PROC TABULATE from Figure 3 is used without the NOSEPS option.

The RTF escape syntax (\R/RTF) and the control word (“\line ”) are added to the AGECL format in front of the first format value, the tabular result is shown in Figure 13. Note that the empty line is at the beginning of every AGECLASS crossing.

Placing the RTF escape code somewhere else in the PROC TABULATE syntax, for example in the labels within the row table space, did not give any satisfying result. Best example I came up with is Method 6.

Figure 12: Method 6 syntax.

```
PROC FORMAT;
 value agecl 1 = '\R/RTF"\line " Young'
             2 = 'Middle'
             3 = 'Older';
run;

ods rtf file="EM6.rtf";
ods escapechar='\';

PROC TABULATE data=a;
 class gender ageclass tcpf;
 format gender $sex.
        ageclass agecl.
        tcpf tcpf.;
  table (gender=' ' All='All' ) * (ageclass=' '),
        (tcpf='Treatment Group')*
        (n='n'*f=5.0 pctn<ageclass>='%'*f=5.1)
  /rts=30 box=' ';
run;

ods rtf close;
```

Figure 13: RTF output of Method 6.

		Treatment Group					
		Group A		Group B		Group C	
		n	%	n	%	n	%
Female	Young	3	75.0	1	20.0	2	66.7
	Middle	1	25.0	1	20.0	1	33.3
	Older			3	60.0		
Male	Young	1	50.0	1	50.0	2	100.0
	Middle			1	50.0		
	Older	1	50.0				
All	Young	4	66.7	2	28.6	4	80.0
	Middle	1	16.7	2	28.6	1	20.0
	Older	1	16.7	3	42.9		

CONCLUSION

Creating empty lines in tables can be done in different ways using PROC TABULATE as the methods show. Still, every method has its own features and the user can use the different methods as a handle for his/her own tabular design.
For Method 1 and 3 the RTF formatted document give a neat empty line, although this empty line is not an added value following tabular output like in Figure 10.
If one wants to introduce empty lines in an RTF formatted document, it is recommended to introduce the empty line within the PROC TABULATE syntax instead of the ODS RTF facility.

REFERENCES

(1) Lex Jansen, Creating PDF® Documents Including Links, Bookmarks and a Table of Contents with the SAS Software, Conference Proceedings, PharmaSUG 2001
(2) Haworth, Lauren E. 1999. PROC TABULATE by Example, Cary, NC: SAS Institute Inc., 1999
(3) SAS Guide to TABULATE Processing, second edition, copyright 1990 by SAS Institute Inc., Cary, NC USA
(4) Haworth, Lauren E., Output Delivery System: The Basics, Cary, NC: SAS Institute Inc., 2001

ACKNOWLEDGEMENTS

I would like to thank Lex Jansen, Gertjan van Maaren and Kit Roes for the support and the review of this paper. Furthermore, everyone within our Biometrics department who helped or gave tips regarding this topic.

CONTACT INFORMATION

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

Twan van Berkel
NV Organon
Department of Clinical Information Oss / Biometrics
Post-box 20
Oss, 5340 BH
The Netherlands
Work Phone : +31 (0) 412 661205
Fax : +31 (0) 412 662516
Email: t.vanberkel@organon.com

APPENDIX 1

Table created with ODS RTF by using the Method 2 syntax.

Table C			Treatment Group					
			Group A		Group B		Group C	
			n	%	n	%	n	%
Female	Young		3	75.0	1	20.0	2	66.7
	Middle		1	25.0	1	20.0	1	33.3
	Older				3	60.0		
Male								
	Young		1	50.0	1	50.0	2	100.0
	Middle				1	50.0		
	Older		1	50.0				
All								
	Young		4	66.7	2	28.6	4	80.0
	Middle		1	16.7	2	28.6	1	20.0
	Older		1	16.7	3	42.9		

APPENDIX 2

Table created with ODS RTF by using the Method 4 syntax.

Example 2				Treatment Group		
				Group A	Group B	Group C
	Gender	Female	- N	4	5	3
			- Mean	24.5	56.8	28.3
			- Minimum	14.0	15.0	15.0
			- Maximum	41.0	80.0	51.0
		Male	- N	2	2	2
			- Mean	41.0	27.0	22.0
			- Minimum	22.0	13.0	19.0
			- Maximum	60.0	41.0	25.0
	Age	Young	- N	4	2	4
			- Mean	19.8	14.0	19.5
			- Minimum	14.0	13.0	15.0
			- Maximum	25.0	15.0	25.0
		Middle	- N	1	2	1
			- Mean	41.0	38.0	51.0
			- Minimum	41.0	35.0	51.0
			- Maximum	41.0	41.0	51.0
		Older	- N	1	3	
			- Mean	60.0	78.0	
			- Minimum	60.0	74.0	
			- Maximum	60.0	80.0	