Paper PO06

Inline Formatting

Duong Tran, Independent Consultant, London, Great Britain

ABSTRACT

With the existing of the Output Delivery System (ODS) from SAS® v8.2 onwards, report can now be much more elaborated. This paper discusses how this can be achieved partly with inline formatting.

In my opinion there are still confusions on this topic within the SAS users' community and I believe is due to the lack of documentation and demonstrated examples from the SAS Institute.

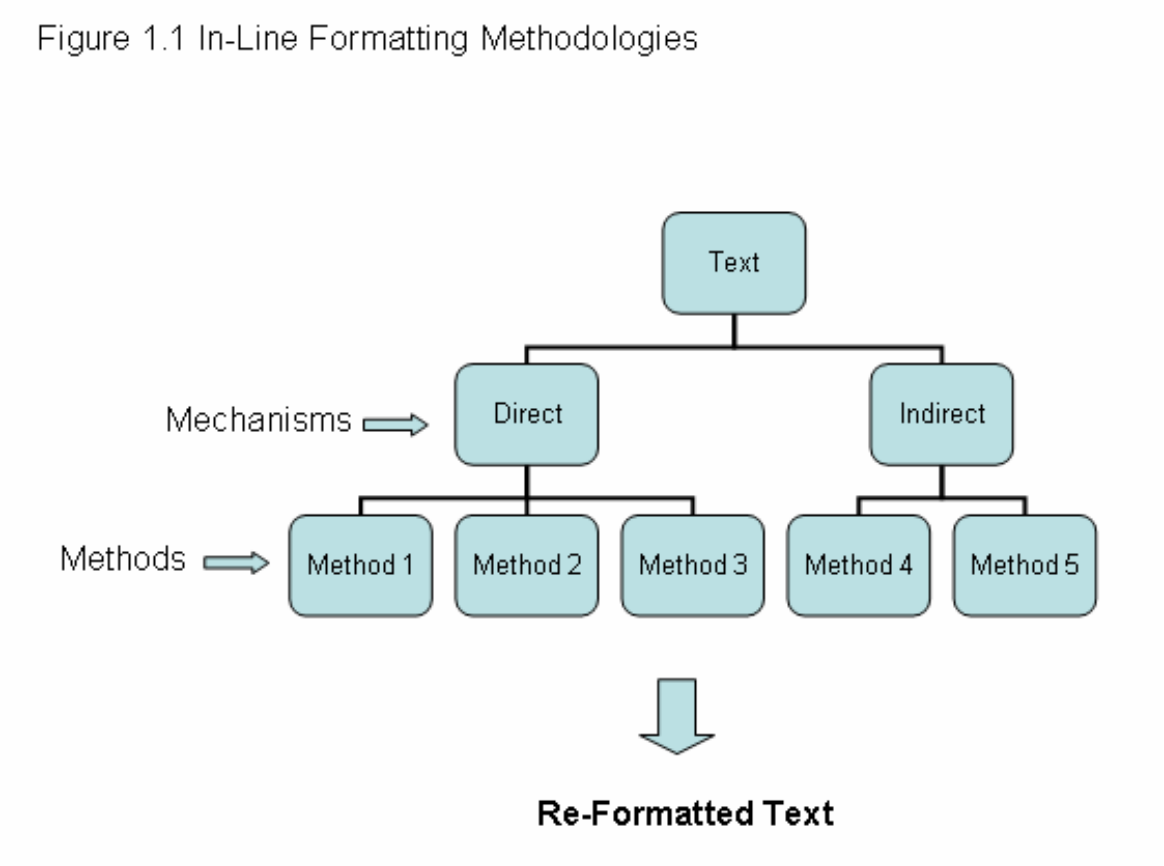
This paper will explain 'All' regarding inline formatting under the ODS RTF (Rich Text Format) destination.

INTRODUCTION

It is almost certain that by opting an ODS destination such as RTF that one would be looking to take advantage of the RTF format, for example chose a font style to emphasis the titles, use super subscript for referencing etc. You can do this with a technique known as inline formatting. SAS provides two mechanisms for this technique and within each mechanism there are several methods. These are summarised in figure 1.1 and will be discussed in this paper in the context of the ODS RTF destination.

INLINE FORMATTING METHODS

Figure 1.1 In-Line Formatting Methodologies



MECHANISM 1 [DIRECT]

Here you insert native RTF commands in the text string directly. In RTF the commands are the control words separate by the backslash.

Example 1

String = “This is direct mechanism \fs50 OK”;

This mechanism is ODS destination dependent, that is \fs50 is a valid command (change font size to 50) in RTF but not in other destinations like PDF or HTML etc.

MECHANISM 2 [INDIRECT]

Here SAS provides functionality to re-format the text string. This mechanism is more generic and works for different ODS destinations.

Example 2

String = “This is indirect mechanism ^S={font_size = 50} OK ^S={}”;
String = “^{dagger} This is the dagger symbol”;

SAS will translate this appropriately according to the ODS destination you specified. Note the ^S={} reset the style.

ESCAPECHAR AND PROTECTSPECIALCHAR

In order recognise what are commands text and what are literal text there has to be a way for SAS to distinguish these and SAS does this by means of an escape character and the option protectspecialchars.

ESCAPECHAR

This is a character you choose to identify S={*style-attributes*} where S is the style function and *style-attributes* are the style attributes to be applied to the text string or ^{*functions-name*}, where *functions-name* are the inline formatting functions (super, dagger etc)

Example 3

Ods escapchar = “^”; /* escape character */
String = “This is indirect mechanism ^S={font_size = 50} OK ^S={}”;
String = “This is the dagger symbol ^{dagger}”;

PROTECTSPECIALCHAR

This can take a value of **auto/on/off** and you can specify this at the location you intend for this to be applied to. By setting protectspecialchars = off, any valid RTF commands will interpreted as commands and not literally. So in the next example, \fs50 is taken as a command (i.e. font size 50).

Example 4

```
Ods listing close;

Ods rtf file=”test.rtf” style = minimal bodytitle;

Proc report data=sashelp.class nowd style(header) = {protectspecialchars = off};
Column age;
Define age / display style(header) = {just=left} “This is direct mechanism \fs50 OK”;
Run;

Ods rtf close;
Ods listing;
```

Note in this example the protectspecialchars = off is only set at the header location. Other locations will take the default value and this could be auto/on/off depends on how you set these in your template style.

METHOD OF INLINE FORMATTING FOR MECHANISM 1

METHOD 1

This method involves setting protectspecialchars = auto and the uses of {}s. These {}s must be correctly used, that is the whole text string must be bounded by {}s.

Example 5

```
/* cw represent some control words          */
/* this is OK                               */
String = “{\cw contents}”;

/* composites like this are not allowed     */
String = “{\cw contents}{\cw contents}” ;

/* but if the composites are bounded as a whole like this then it is OK */
String = “{{\cw contents}{\cw contents}}”;

Ods listing close;

Proc template
  Define style newstyle
     Parent = styles.minimal
     Style systemtitle from systemtitle / protectspecialchars = auto;
  End;
Run

Ods rtf file=”test.rtf” style = newstyle bodytitle;
Title1 “{{Remember the parent braces }{must bound the whole string}}”;
Title2 “{needs proctectspecialchars = auto \line\fs30 but escapechar is not needed}”;
Proc print data=sashelp.class;
Run;

Ods rtf close;
Ods listing;
```

Note if protectspecialchars = off, the above example will also works as meant to be but the {}s would be redundant in this context.

METHOD 2

In this method you need to set protectspecialchars = off at the locations required.

Example 6

```
Proc template;
  Define style newstyle;
     Parent = styles.minimal;
     Style systemtitle from systemtitle / protectspecialchars = off;
  End;
Run;

Ods listing close;
Ods rtf file=”test.rtf” style = newstyle bodytitle;

Title1 “This is line 1 \line This is line 2”;

/* note the {}s are redundant in this context */
Title2 “This is line 1 {\line} This is line 2”;
```

```
/* note the {}s are not redundant in this context because \ul\fs30 is applicable to */
/* the first section of the string only.                                            */
Title3 “{\ul\fs30 This is line 1} \line This is line 2”;

Proc report data = sashelp.class nowd style(header) = {protectspecialchars = off};
   Column sex age;
   Define sex / display      style(header column)  = {just = left cellwidth=50%} “Patient \line\ul Gender”;
   Define age / display      style(header column)  = {just = left cellwidth=50%} “Patient \line\ul Age”;
Run;

Ods rtf close;
Ods listing;
```

Note the setting of protectspecialchars = off for the titles is done via proc template. This can also be done through the style function as in the next example.

Example 7

```
Ods listing close;
Ods escapchar = “^”;

Ods rtf file=”test.rtf” style = minimal bodytitle;

Title1 “^S={proctectspecialchars = off} \fs20\ul switching proctectspecialchars to off”;
Title2 “This usage is ditto for footnotes”;

Proc print data=sashelp.class;
Run;

Ods rtf close;
Ods listing;
```

METHOD 3

This method use the escape character and the R/RTF command in the form of ^R/RTF’raw-text’ where ^ is the escape character, R the raw-text function and /RTF means that the raw text is passed only to RTF. Note in this method proctectspecialchars is not involved.

Example 8

```
Ods listing close;
Ods escapechar = “^”;

Ods rtf file=”test.rtf” style = minimal bodytitle;

/*
note there must be at least one space between the end of last control word and the right hand side speech mark in
this sequence of text ^R/RTF’\fs30 ‘
*/
Title1 “This is a test ^R/RTF’\line\fs30 ‘OK ^R/RTF’\fs ‘Back to default font size”;
Title2 “Note ^R/RTF’\fs40 ‘a space at the end of the control words sequence”;

Proc print data=sashelp.class;
Run;

Ods rtf close;
Ods listing;
```

METHOD OF INLINE FORMATTING FOR MECHANISM 2

In the methods for this mechanism you will need to use an escape character.

METHOD 4

In this method you use the style-attributes similar to you would use the style attributes in proc report etc.

Example 9

```
Ods listing close;
Ods escapechar = "^";

Ods rtf file="test.rtf" style = minimal bodytitle;
Title1 "This is indirect mechanism ^S={font_size = 30} OK ^S={} Back to default font size";

Proc print data=sashelp.class;
Run;

Ods rtf close;
Ods listing;
```

METHOD 5

In this method you use the inline formatting functions shown below for v8.2 and v9.1.3. See later for a more open-ended user-configurable in v9.2.

```
{super text}    /* put text in superscript                          */
{sup text}      /* put text in subscript                            */
{dagger}        /* the dagger character                             */

Also in SAS v9
{thispage}      /* the current page                                 */
{numpages}      /* number of page in the document                   */
{pageof}        /* this same as using ^{thispage} of ^{numpages}    */
```

Example 10

```
Ods listing close;
Ods escapchar = "^";

Ods rtf file="test.rtf" style = minimal bodytitle;
Title1 "^{dagger}This is the dagger symbol";

Proc print data=sashelp.class;
Run;

Ods rtf close;
Ods listing;
```

SOME SUBTLETY DIFFEERENCE BETWEEN THE INLINE FORMATTING METHODS

Some usage of these methods has a very subtle difference but the implications in the background are quite different.

Example 11

```
Proc template;
  Define style newstyle;
     Parent = styles.minimal;
     Style systemtitle from systemtitle / protectspecialchars = off;
  End;
Run;
```

```
Ods listing close;
Ods escapechar = “^”;
Ods rtf file=”test.rtf” style = newstyle bodytitle;

title1 “^{super a} This is an indirect method and generically works for different ODS destinations”;
title2 “{super a} This has the wrong syntax for super subscript”;
title3 “This is a direct method but the braces are redundant in this context {\super a}”;

Proc print data=sashelp.class;
Run;

Ods rtf close;
Ods listing;
```

Example 12

This example use the symbol font to get the ≥ symbol.

```
Ods escapechar="^";
Ods rtf file='test.rtf' style=minimal bodytitle;

title 'This shows the special character in the label';

proc print data=sashelp.class label;
label age="Age" "^S={font_face=symbol}" "B3"x "^S={}" "10";
run;

Ods rtf close;
```

You can use the symbol font to get Greek’s alphabet or symbols etc. but this technique is quite cumbersome. A better technique is to use Unicode.

Example 13

```
Proc template;
  Define style newstyle;
     Parent = styles.minimal;
     Style systemtitle from systemtitle / protectspecialchars = off;
  End;
Run;

Ods listing close;
Ods rtf file=”test.rtf” style = newstyle bodytitle;

title1 “You can find the unicodes at www.unicode.org/charts”;
/* note you will need at least two spaces at the end of the unicode */
title2 “This is the code \u8734  for the infinity symbol”;
/*
this will not work though
title3 "Note this is not OK \u8734  ";
*/

/* note if the unicode is the last word in the string then it must be bound with {}s */
title3 "Note this is OK {\u8734  }";
/* but if you use the raw function then {}s are unnecessary but still need the two spaces */
title4 "Note this is OK also ^R/RTF'\u8734  '";

Proc print data=sashelp.class;
Run;
```

```
Ods rtf close;
Ods listing;
```

Note this example demonstrate the subtleness when using Unicode. All other rules explained previously should be applicable here also.

Example 14

There are confusions in the SAS users' community still regarding the usage of the escapechar. This example shows the typical confusion.

```
Proc template;
  Define style newstyle;
     Parent = styles.minimal;
     Style systemtitle from systemtitle / protectspecialchars = on;
  End;
Run;

Ods listing close;
Ods escapechar="^";
Ods rtf file="test.rtf" style = newstyle bodytitle;

/* This has a wrong syntax and should not work but do in v9.1.3 so I think it is a bug */
title1 "Under ^{\super SAS v9.1.3} this works but not under v8.2";    /* note the backslash */
Print data=sashelp.class;
Run;

Ods rtf close;
Ods listing;
```

I believe the correct usage should either be:

a) The direct method

```
title1 "Under {\super SAS v9.1.3} this works - also under v8.2";
with
Style systemtitle from systemtitle / protectspecialchars = off;
```

OR

```
title1 "^S={protectspecialchar=off} Under {\super SAS v9.1.3} this works - also under v8.2";
```

b) The indirect method

```
title1 "Under ^{super SAS v9.1.3} this works - also under v8.2";
```

INLINE FORMATTING FUNCTIONS

There are new features for inline formatting in SAS v9.2 and the general format for this new inline style syntax is:

```
^{functions-name <arg-1 <arg-2 … <arg-n>>>}
```

where "^" is the ODS escape character. Here are some examples of some inline formatting functions:

```
^{dagger}
^{super a}
^{unicode 8734}
^{style [color=green] formmated text}
```

and these can be nested:

```
^{super {unicode 8734}}
```

I have mentioned the subtleness of using unicode directly in example 13 but there is a way to eliminate this subtleness by fooling RTF, that is by introducing a none effective or dummy control word.

```
/* by using this syntax there are no subtleties to worry about – I use this form all the time now */
title1 “By introducing a none effective control word like this {\super{\u8734\fs}}”;
```

So in respect to the ODS RTF destination there is nothing new in the Inline Formatting Functions that could not be done by the direct method. On the contrary I would probably still prefer to use the direct method in SAS v9.2 because in my opinion it is simpler and why use a less general method. Imagine if you want to bold the super subscript in RTF – with the direct method this is just **{\b\super a}**. How would you do this with the new features in SAS v9.2. I have not checked this because v9.2 is still unavailable. Also under v8.2 and v9.1.3 the following example gives an undesirable space before the “)” character. I believe this is a bug hence another reason to use the direct method instead as in a) in example 14.

```
Title1 “Unit = (kg/m^{super 2})”;
```

OTHERS

You may come across these also but I believe the originator of this from the SAS Institute mentioned that these were experimental in v8.2 and that they will not be developed further in later version.

^n, ^-2n
: Inserts a new line. The second form wraps around to the position of the mark if supported.

^w
: Marks a good place for an optional line-break. Doesn't force a new line, but "suggests" it as a good place if the line must be wrapped.

^_
: Inserts a non-breaking space.

^1z, ^2z, ^3z, ^4z
: Error, Warning, Note, and Fatal tags, respectively. Lets the user format error messages in the same way that SAS procedures do.

CONCLUSION

Hopefully this paper has given you the concepts and insights of inline formatting. It is only when these concepts and their subtleness are grasped that you will begin to appreciate why this work here and not there?

I have discuss this paper in the context ODS RTF destination but I believe these principals are the same with other ODS destinations.

REFERENCES

[1] http://support.sas.com/rnd/base/topics/templateFAQ/Template.html
[2] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnrtfspec/html/rtfspec.asp
[3] RTF Pocket Guide (Buke S. M.)

CONTACT INFORMATION

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

Duong Tran
Phone: 07752326413
Email: trand000@aol.com
www.tranz.co.uk