PharmaSUG 2013 - Paper IB08

Pretty Please?! Making RTF Output "Pretty" with SAS

Carol Matthews, United Biosource Corporation, Blue Bell, Pennsylvania
Elena Kalchenko, United Biosource Corporation, Blue Bell, Pennsylvania

ABSTRACT

Output generated from SAS® in the pharmaceutical industry invariably ends up in a document that is presented to regulatory authorities, review boards, publications, and a variety of other audiences. As SAS has evolved to more easily generate output that can be read directly into software packages such as Microsoft Word, the expectation that SAS output will look "published" is becoming the standard. Recipients of SAS output now expect actual superscript "a" as a footnote symbol rather than a regular text "a" or "*" symbol. While some features such as fonts, margins and tabs can be controlled directly with SAS, other features need to be added with RTF code. In this paper, we will discuss the combination of SAS and RTF code that is required to generate output directly from SAS that looks like it was word-processed. We will touch on the basic components of Proc Template needed to control the margins and overall appearance of output, the basic ODS syntax needed to control where output goes and how SAS interacts with RTF, various features of PROC REPORT that can be used to further refine the look of output, and common RTF code that can be used to put the final polish on tables and listings.

INTRODUCTION

One of the most significant additions to SAS has been the ability to automatically generate output in a number of different file formats. Rather than just simple text files, SAS can now create output as a variety of different file formats, including HTML, PDF, and RTF. In the pharmaceutical industry, the results produced by SAS programs are almost always included in Microsoft Word documents, either as in-text tables or as report appendices. As a result, often the most desired file format for document authors is RTF because these files can be easily be copied from the original output file and inserted directly into documents, often with little or no editing by the author. It is important for programmers to understand the needs of document authors so SAS output can be tailored to meet the author's needs.

While SAS does provide a default output style when creating RTF output, often this default needs to be modified to reduce the amount of editing that authors are required to do when including output in documents. To effectively modify the RTF output generated by SAS, programmers need to understand a little about the RTF file format itself. RTF (short for Rich Text Format) is actually just a text file with a defined structure that preserves the formatting of the document when opened in word processing software packages (usually Microsoft Word). SAS version 9.1.3 creates RTF files that can be opened directly in Microsoft Word 2000 or later. SAS 9.2 and 9.3 are both designed to work with Word 2002 or later, although in most cases there is no appreciable difference in RTF code between the different Word versions (RTF code inserted in programs run in SAS 9.1.3 will also run in later versions, usually with identical results).

SAS provides a number of techniques for programmers to create and modify output as RTF files, ranging from simply directing output to an RTF file to controlling the appearance of specific words. Programmers need to understand how to modify general as well as specific aspects of RTF files to create output that document authors can easily include in documents.

ODS BASICS

The first step in creating RTF output for easy use by word processing packages like Microsoft Word is simply to tell ODS that the output being created should go to an RTF file, and to specify the location and name of that file. In most cases, output should only go to this RTF file, and not the output window (in interactive SAS) nor a .LST file (batch SAS). This code should be inserted immediately prior to the procedure (usually PROC REPORT) that will be producing the output intended to go into the RTF file. To "turn off" the default output destination and redirect output to an RTF file, use the following code:

```
ods listing close ;
ods rtf file="C:\output\Table_1.rtf" ;
```

There are SAS system options that are also needed to control the look of RTF output. In most cases, the format and placement of the date and page number will need to be specified, as well as the orientation of the page when it prints. To prevent the automatic printing of the system date and page, use the NODATE and NONUMBER system options. Use the ORIENTATION system option to make output landscape or portrait.

```
options nodate nonumber orientation=landscape ;
```

One other important ODS feature that should be initialized is the ODS ESCAPECHAR. This option will set the symbol that SAS uses to recognize when RTF code is included within your SAS code. Note that a character that is highly unlikely to appear in the data or in the title and footnote text should be selected. In the example below, the escape character is beinge set to the tilde (~) symbol, but the carrot or hat symbol (^) is also a commonly used symbol:

```
ods escapechar='~' ;
```

The final step in creating an RTF file is to close the file once the desired output that should be included is complete. It is also good practice to reset the output destination back to the default (LISTING) and reset the modified system options for the date and page number, as demonstrated by the code below.

```
ods rtf close ;
ods listing ;
options date number ;
```

Essentially, creating an RTF file involves surrounding the code that generates output with ODS statements. A common structure of code is as follows:

```
Code to set up/analyze the data…

options nodate nonumber orientation=landscape ;
ods listing close ;
ods escapechar='~' ;
ods rtf file="C:\output\Table_1.rtf" ;

PROC REPORT code…

ods rtf close ;
ods listing ;
options date number ;
```

An example program that will be modified throughout this paper is shown below. Note that for the purposes of this paper, the orientation of the page has been left as portrait for easier inclusion of output examples.

```
options nodate nonumber ;
ods listing close ;
ods escapechar='~' ;
ods rtf file="C:\Output\Table 1.rtf" ;

title1 "Table 1" ;
title2 "Medical History" ;
title3 “(Safety Population)”;
footnote "Note: Patients are counted at most once for each body system. " ;

proc report data=numbers missing nowindows center split='|' ;
   column bsysid bodysys ("Total Patients|(N=20)" cnt pct) ;
   define bsysid   / order noprint ;
   define bodysys  / "Body System" ;
   define cnt      / "n" ;
   define pct      / "(%)" right ;
   compute after bsysid ;
      line  " " ;
   endcomp ;
run ;
ods rtf close ;
ods listing ;
options date number ;
```

The output that results from running the code above is show in Output 1 below.

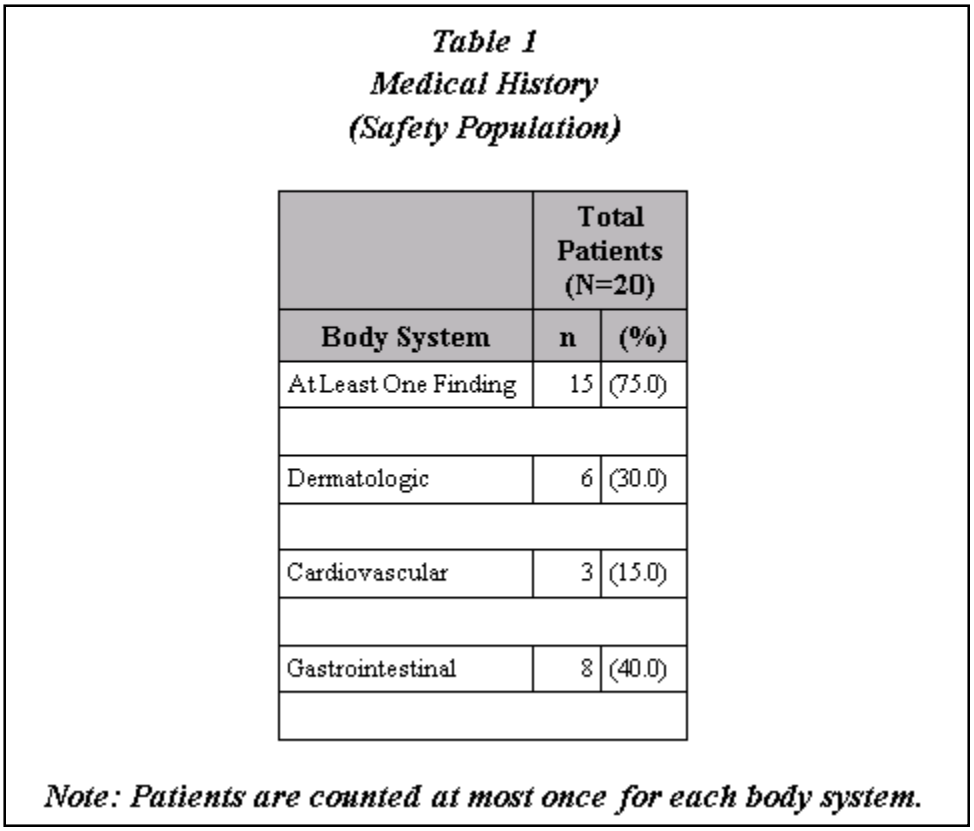

Table 1
Medical History
(Safety Population)

	Total Patients (N=20)	
Body System	**n**	**(%)**
At Least One Finding	15	(75.0)
Dermatologic	6	(30.0)
Cardiovascular	3	(15.0)
Gastrointestinal	8	(40.0)

Note: Patients are counted at most once for each body system.

Output 1: Default RTF Style

Once SAS is set to send the program output to RTF, there are additional output characteristics that will likely need to be modified to make the output acceptable for inclusion in larger documents. These characteristics can be specified in report templates that are controlled using the PROC TEMPLATE procedure.

CREATE A CUSTOM PAGE STYLE WITH PROC TEMPLATE

As shown in the example in the previous section, the default RTF output style is likely not ideal for inclusion in reports. The next step in creating RTF output that *can* be easily used in documents is setting up the general document properties – margins, fonts, general table, header and footer characteristics. The best way to guarantee a uniform look across all output is to create a single document template that is used when generating all final output. PROC TEMPLATE provides the means to accomplish this. While there are many advanced aspects of SAS output that can be controlled through PROC TEMPLATE, the code below constitutes the "basics" required for producing RTF output for inclusion in documents.

MODIFY THE DEFAULT RTF STYLE

SAS provides a default style that is automatically used when the ODS RTF statement executes. This default RTF style that can be modified to produce output with different attributes as required. Start by assigning a name to the modified RTF file with the DEFINE statement, and basing the new style on the default style provided by SAS with the PARENT statement.

```
proc template ;
  define style MyStyleRTF ;
    parent=styles.rtf ;
```

Aspects of this new style can now be modified as necessary to meet the requirements of whatever document in which the output will be included. Note that there are many aspects of this style, most of which do not need to be modified. The key elements that are most likely to need to change will be described here.

The first style element that will likely need to be modified is the font used throughout the RTF file. The default style uses a variety of fonts, font sizes and attributes (bold, italics). These will likely need to be changed to match the

requirements of the person or group who will be including the output in or with other documents. The *fonts* element is can be modified to change the fonts used within the document. The code below changes the fonts for the various parts of the RTF output to Times New Roman, specifies all of the font sizes, and changes the font attributes to bold or italic as required.

```
replace fonts /
  'TitleFont'     = ("Times New Roman",10pt,Bold) /* TITLE statement */
  'TitleFont2'    = ("Times New Roman",10pt,Bold) /* PROC titles */
  'headingFont'   = ("Times New Roman",10pt,Bold) /* Table column/row headings */
  'docFont'       = ("Times New Roman",10pt)      /* data in table cells */
  'footFont'      = ("Times New Roman",8pt)       /* FOOTNOTE statements */
  'StrongFont'    = ("Times New Roman",10pt,Bold)
  'EmphasisFont'  = ("Times New Roman",10pt,Italic)
  'headingEmphasisFont'= ("Times New Roman",10pt,Italic)
  'FixedFont'          = ("Times New Roman",10pt)
  'FixedEmphasisFont'  = ("Times New Roman",10pt,Italic)
  'FixedStrongFont'    = ("Times New Roman",10pt,Bold)
  'FixedHeadingFont'   = ("Times New Roman",10pt)
  'BatchFixedFont'     = ("Times New Roman",10pt) ;
```

The next aspect of the output that will likely need to change is the cell borders. The default style shows a solid line border around all cells in the body of the table. While in some cases this may be desirable, it is not usually the preferred look for end-of-text tables. In the example below, the *rules=groups* removes the solid lines around all of the cells but leaves only the row heading cells showing a solid line on the bottom border. Spacing within and around the cells is removed to allow more content to show on each page, and the width of the border on the top of the table is set to be thick. Each of these elements has multiple options that are described in the SAS documentation, as well as several SAS Books by Users.

```
style table from table /
  background=_UNDEF_ /* REMOVES TABLE BACKGROUND COLOR */
  rules=groups       /* INTERNAL BORDERS: SET TO BOTTOM BORDER ON ROW HEADERS */
  frame=above        /* EXTERNAL BORDERS: SET TO TOP LINE OF TABLE ONLY */
  cellspacing=0      /* SPACE BETWEEN TABLE CELLS */
  cellpadding=0      /* REMOVES PARAGRAPH SPACING BEFORE/AFTER CELL CONTENTS */
  borderwidth=1.5pt; /* SET WIDTH OF BORDER IN FRAME= */
```

To remove the shading from the column heading rows and change the font, the *HeadersAndFooters* style element also needs to be updated. The code below removes the background shading and makes the font bold per the *fonts* element specified earlier.

```
replace HeadersAndFooters from Cell /
  background = _undef_
  font = Fonts('TitleFont') ;
```

Text in FOOTNOTE statements default to using the same font as TITLE statements and are centered on the page. To change the font to the new "footFont" that is specified in the *fonts* style element, the *SystemFooter* element must be updated as shown below. This code also left-justifies the footnote text.

```
replace SystemFooter from TitlesAndFooters /
   font = Fonts('footFont')
   just = LEFT ;
```

The final aspect of the document that will likely need to be modified is the page margins. The default margins are very close to the edge of the page and generally published output requires at least a one-inch margin all around the page. Note that when margins are set, the actual distance on a printed page as measured with a ruler may be slightly different than what is specified for the top and bottom of the page (left and right margins are accurate). The code below changes the margins to 1 inch on all sides but the bottom, but when printed and measured the ¾ inches for the bottom margin actually measures closer to one inch.

```
replace Body from Document /
  bottommargin = .75in
  topmargin    = 1in
  rightmargin  = 1in
  leftmargin   = 1in ;
```

```
  end ;  /* END OF DEFINE STYLE MyStyleRTF */
run ;    /* END OF PROC TEMPLATE */
```

In the code above, the original DEFINE statement requires an END statement once all of the individual elements are finished being modified. The RUN statement finishes the procedure. When the program that contains this PROC TEMPLATE code is run, a new template named MyStyleRTF will be created which can subsequently be used by ODS when RTF output is generated.

STORE AND USE CUSTOM TEMPLATES

Now that default report template has been modified, it can be stored it in a permanent catalog so many programs can refer to it rather than needing to recreate it in each SAS session. This is done by adding a LIBNAME statement to indicate the location to store the template, then setting the ODS PATH *prior* to the PROC TEMPLATE code. In parentheses after each template reference, the access permissions are set. This needs to be set to "UPDATE" for user-defined templates if users can be allowed to modify these templates.

```
libname optlib "C:\myfiles" ;
ods path optlib.templat(UPDATE) sashelp.tmplmst(READ) ;
```

To access the new user-defined template from other programs, a LIBNAME statement that points to the location of the user-defined template library is required. Once that is set, the following ODS PATH statement can be added to any program that needs to access these user-defined templates. Note that the user-defined template library is referenced first, followed by the default templates provided with SAS. This is to ensure that the user-defined templates are used if they exist, and if not then SAS will use the default instead. Access to both is also set to "READ" so programmers do not accidentally modify the templates in other programs.

```
ods path optlib.templat(READ) sashelp.tmplmst(READ) ;
```

With the new template created and stored, the final step is to tell ODS to use the user-defined template rather than the system default template. This is done in the ODS statement by adding the STYLE= option.

```
ods rtf style=MyStyleRTF file="C:\output\Table 1.rtf" ;
```

The previous example output looks like the sample in Output 2 below when the custom style described above is applied. Note that the table titles ("Table 2" and "Medical History") are added using TITLE statements and the footnote is added with the FOOTNOTE statement.

Table 1
Medical History
(Safety Population)

Body System	Total Patients (N=20) n (%)
At Least One Finding	15(75.0)
Dermatologic	6(30.0)
Cardiovascular	3(15.0)
Gastrointestinal	8(40.0)

Note: Patients are counted at most once for each body system

Output 2: Custom RTF Style

While modifying the default RTF template can change the appearance of output dramatically, there are even more aspects of this file format that can be modified to make output ready to “cut and paste” directly into reports and other documents. PROC REPORT provides a number of ways to manipulate RTF output even further.

PROC REPORT TECHNIQUES

There are several ways to add additional formatting changes to output created by PROC REPORT. The STYLE option can control the format of specific columns and COMPUTE blocks can be used to control where text appears in output.

One important aspect to understand when generating output in RTF is where the titles and footnotes actually appear in the document. There is a difference between *page* titles and footnotes (controlled by the TITLE and FOOTNOTE statements) and *table* titles and footnotes (controlled by the COMPUTE block). Page titles and footnotes appear in the Word header and footer sections of the document and appear the same on every page. Table titles and footnotes are part of the table itself and may not necessarily appear on every page (for example, a table footnote may only appear on the last page of the table). It is important to discuss with document authors where they want titles and footnotes to appear. In the original example program, the Medical History table titles and footnotes were created with TITLE and FOOTNOTE statements so appear as part of the page header and footer. To change the location so the titles and footnote are included within the table itself, the compute block is used instead.

```
compute before _page_ ;
   line "Table 1" ;
   line "Medical History" ;
   line "(Safety Population)" ;
   line ' ' ;
endcomp ;

compute after _page_ ;
   line "Note: Patients are counted at most once for each body system." ;
endcomp ;
```

In addition to moving the location of the titles and footnotes, it is possible to adjust the width of the columns so they appear more appropriately spaced for viewing. The STYLE option can set the width of columns in inches. It can also be used to change the alignment of a column, as shown for the *cnt* variable below.

```
define bodysys  / "Body System" style=[cellwidth=2in] ;
define cnt      / "n"           style=[cellwidth=.5in
                                       protectspecialchars=OFF
                                       pretext="\qr "] ;
define pct      / "(%)"         style=[cellwidth=.5in] ;
```

When the new code is executed, the output looks like Display 3 below.

Table 1
Medical History
(Safety Population)

Body System	Total Patients (N=20) n	(%)
At Least One Finding	15	(75.0)
Dermatologic	6	(30.0)
Cardiovascular	3	(15.0)
Gastrointestinal	8	(40.0)

Note: Patients are counted at most once for each body system.

There are a number of other useful tools in RTF that can be used to control how output is displayed in specific columns. Several of these are shown in Table 1.

Function	PROC REPORT Code
Left-justify a single column (including the column header)	define *varname* / "*header text*" style(column) = [protectspecialchars=OFF **pretext="\ql "**] ;
Right-justify a single column (including the column header)	define *varname* / "*header text*" style(column) = [protectspecialchars=OFF **pretext="\qr "**] ;
Set tab stops, where # is the distance from the margin in twips[a]	define varname / "*header text*" style(column) = [protectspecialchars=OFF pretext="*tab stops you want* "] o left-flush tab: \tx# o right-flush tab: \tqr\tx# o center tab: \tqc\tx# o decimal tab: \tqdec\tx#
Align text at the bottom of a cell	define *varname* / "*header text*" style(column) = [vjust=bottom];
Set the width of a column, where #.# is the width in inches	define *varname* / "*header text*" style(column) = [cellwidth=#.#in];

[a] *There are 1440 twips in an inch – each tick mark on the Word ruler is 180 twips.*

Table 1: Useful PROC REPORT Code

Just as PROC REPORT options can further refine the look and structure of RTF output, there are still other ways to control very specific aspects such as the font and alignment of individual cells. Some knowledge of RTF code is required to achieve this level of control, but it is a relatively simple language where a little information can go a long way toward making final output ready to copy and paste into documents.

RTF CODE CAN FURTHER REFINE OUTPUT

While PROC REPORT can be used to control specific aspects of rows, columns and page breaks, there are situations where finer control of text is required. Making the text of a specific column heading left-justified while others remain centered, adding superscript text to indicated footnotes, and making the text in select cells bold font are all examples where more specific control of text is needed. This type of formatting can be accomplished by including RTF code within the values of text strings or character variable values.

In the Medical History Table example that has been shown throughout this paper, the final step in formatting this output is to make the first line of the table title bold, make the body system column heading left-justified, add a line over the footnote cell, "clean up" the footnote text presentation and make the text of the "At Least One Finding" row bold font. The first task is to add RTF code to the value of the *bodysys* variable to make the text bold, as shown below.

```
data numbers ;
   length bodysys $200 ; /* MAKE LENGTH LONGER TO INCLUDE RTF CODE */
   set numbers ;
   if substr(bodysys,1,2) eq 'At' then
      bodysys = "~R'\b '" || strip(bodysys) ; /* MAKE BOLD */
run ;
```

Once the data is prepared for PROC REPORT, the next step is to add RTF code to the column heading text in PROC REPORT as set by the DEFINE statement for the *bodysys* variable. The code below makes the column heading text left-justified.

```
define bodysys  / order "~R'\ql 'Body System" ;
```

The next step is to make the first row of the table title bold font. Note that if only *some* of the text in the compute block should be bold font, the bold font needs to be "turned off" at the end of the text string that should be bold.

```
compute before _page_ ;
   line "~R'\b 'Table 1~R'\b0 '" ;
   line "Medical History" ;
   line "(Safety Population)" ;
   line ' ' ;
endcomp ;
```

The final step is to add a top border to the footnote cell (including specifying the cell border width as 30 twips) and adjust the presentation of the footnote so that it is left justified and the second line is indented when it wraps. Notice that several RTF commands can be strung together to impact multiple aspects of a cell.

```
line "~R'\brdrt\brdrs\brdrw30\ql\li500\fi-500 'Note: Patients are counted at
                                               most once for each body system." ;
```

The final output after adding this RTF code is shown in Display 4 below.

Table 1
Medical History
(Safety Population)

Body System	Total Patients (N=20) n	(%)
At Least One Finding	15	(75.0)
Dermatologic	6	(30.0)
Cardiovascular	3	(15.0)
Gastrointestinal	8	(40.0)

Note: Patients are counted at most once for each body system.

Display 4: Refined RTF Output

The example above shows just a few of the specific items that can be controlled by adding RTF code to SAS code. While there are large manuals on the RTF language, the table below lists a number of the most useful items. Note that the code below assumes that the ODS ESCAPECHAR is set to tilde (~) and the entire string is enclosed in double quotes when included in the definition of the value of a character variable or text string.

Function	RTF Code
Add "Page x of y" in page titles/footnotes	Page ~{pageof}
Add a tab in text (note: this does not set the tab stop location, it is the equivalent of hitting the Tab key)	~R'\tab '*text*
Add a hard carriage return	~R'\par ' or ~R'\line '
Make text superscript	~{super *text*}
Make text subscript	~{sub *text*}
Underline text	~R'\ul '*text* ~R'\ul0 '
Make text bold	~R'\b '*text* ~R'\b0 '
Make text italic	~R'\i '*text*~R'i0 '
Center text in a specific cell	~R'\qc '*text*
Left justify text in a specific cell	~R'\ql '
Add a hanging indent where the second and subsequent lines are indented (useful for footnotes); note that 300 is just an example and represents distance in twips[a]	~R'\li**300**\fi-**300** '*text…*
Add the greater than or equal to sign (≥)	~R'\ {\uc2\u8805 >=} *-or-* ~S={font_face=Symbol}"||"B3"x||"~S={}"
Add the less than or equal to sign (≥)	~R'\ {\uc2\u8804 <=} *-or-* ~S={font_face=Symbol}"||"A3"x||"~S={}"
Add the bottom border to a single cell, where # is the width of the border in twips[a]	~R'\brdrb\brdrs\brdrw# '*text*
Add the top border to a single cell, where # is the width of the border in twips[a]	~R'\brdrt\brdrs\brdrw# '*text*
[a] *There are 1440 twips in an inch – each tick mark on the Word ruler is 180 twips.*	

Table 2: Useful RTF Code

CONCLUSION

SAS now provides an easy way to generate output in multiple file formats, one of the most useful being RTF. SAS has given programmers a variety of tools to manipulate the default style of RTF output such that output can be cut and paste directly into Microsoft Word documents with little or no editing. While it may seem that the burden of "word processing" has been moved out of writers' laps and dumped on programmers' desks, the process of getting data into reports is much more efficient when programs can generate "ready to use" output. Gone are the days where document authors spend hours transcribing information from printed SAS output into document tables. Using the techniques described here, programmers can work with document authors to create a style for RTF output that is easy for programmers to create and easy for document authors to use.

REFERENCES

Haworth, Lauren. 2004. "SAS® with Style: Creating your own ODS Template for RTF Output." Proceedings of the Twenty-Ninth Annual SAS® User's Group International Conference, Paper 125-29. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi29/125-29.pdf.

Haworth, Lauren. 2011. "ODS RTF: the Basics and Beyond." Proceedings of the SAS® Global Forum 2011 Conference, Paper 263-2011. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings11/263-2011.pdf

Tong, Cindy. 2003. "ODS RTF: Practical Tips." Northeast SAS Users Group Inc. Sixteenth Annual Conference Proceedings, Paper AT007. Available at http://www.nesug.org/proceedings/nesug03/at/at007.pdf

CONTACT INFORMATION

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

Name: Carol Matthews
Enterprise: United Biosource Corporation
Address: 920 Harvest Drive, Suite 200
City, State ZIP: Blue Bell, Pennsylvania 19422 USA
Work Phone: +1 215 390 2236
E-mail: carol.matthews@unitedbiosource.com

Name: Elena Kalchenko
Enterprise: United Biosource Corporation
Address: 920 Harvest Drive, Suite 200
City, State ZIP: Blue Bell, Pennsylvania 19422 USA
Work Phone: +1 215 390 2241
E-mail: elena.kalchenko@unitedbiosource.com