

Some Techniques for Integrating SAS® Output with Microsoft Excel Using Base SAS®

Vincent DelGobbo, SAS Institute Inc., Cary, NC

ABSTRACT

This paper explains some techniques to integrate your SAS® output with Microsoft Excel. The techniques that are presented in this paper require Base SAS® 9.1.3 SP4 and above, and can be used regardless of the platform on which SAS is installed. You can even use them on a mainframe! Creating and delivering your workbooks on-demand and in real time using SAS server technology is discussed. Although the title is similar to previous papers by this author, this paper contains new and revised material not previously presented.

INTRODUCTION

This paper discusses three techniques for using Base SAS to get data into Excel. Technique #1 produces a comma-separated value (CSV) file that contains only data and no formatting of any kind. Technique #2 produces a CSV file with data and some grouping, but still no formatting and color. Technique #3 produces the workbook in Figure 1, complete with formats, colors, and correct groupings. The paper concludes with a discussion of how to drive dynamic data to Excel using SAS server technology.

<i>Subject ID: 1</i>					
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
-2.00	16	105	135	84	
-1.00	9	59	128	73	
0.02	9	64	126	83	
0.08	9	64	133	91	
0.17	12	72	136	90	
0.25	14	60	130	82	
1.00	16	73	135	73	
2.00	12	72	124	65	
3.00	12	81	132	77	
4.00	21	80	120	70	
5.00	19	80	120	70	
8.00	16	96	150	90	
24.00	20	80	120	70	
<i>Subject ID: 2</i>					
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
-2.00	15	72	94	56	
-1.00	12	55	135	78	
0.02	12	50	127	77	
0.08	14	58	104	73	
0.17	13	54	123	79	
0.25	18	55	122	75	
1.00	24	60	116	78	

Figure 1. Multi-Sheet Excel Workbook Generated Using Base SAS Software

The worksheets contain fictional vital signs data for clinical trials "ABC 123" and "XYZ 987". The worksheets contain one table for each patient enrolled in the trial, resulting in multiple tables in each worksheet. An Excel format, not a SAS format, is used to control the appearance of the **Vital Signs: Time** column, and a TITLE statement is used to display the patient's ID number.

You can download a copy of the code and data used in this paper from the SAS Presents Web site at support.sas.com/saspresents. Find the entry "Some Techniques for Integrating SAS® Output with Microsoft Excel Using Base SAS®".

You cannot use the techniques described in this paper to update existing workbooks; the entire Excel document is created on each execution, and existing workbooks cannot be altered.

The code in this paper was tested using SAS®9.3, version 1.122 of the ExcelXP tagset (shipped with SAS®9.3), and Microsoft Excel 2010 software.

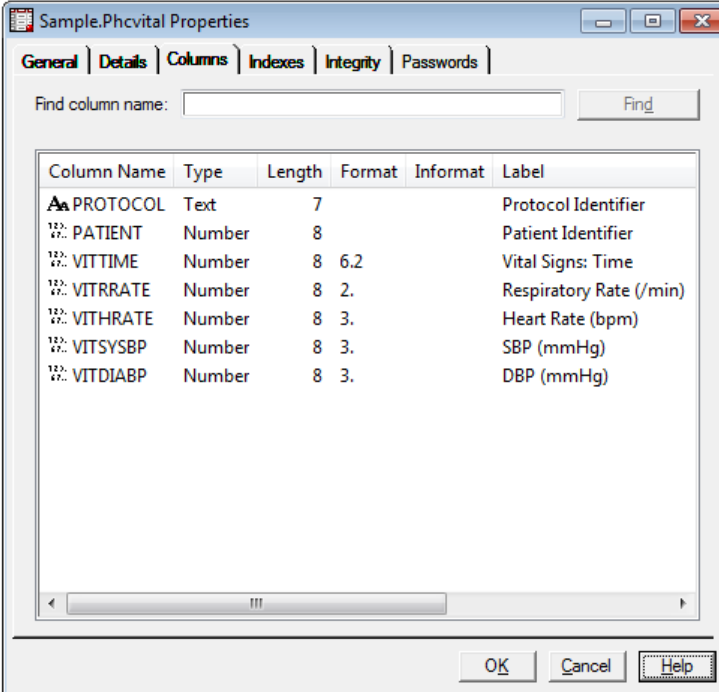
REQUIREMENTS

To use the techniques described in this paper, you must have the following software:

- Base SAS 9.1.3 Service Pack 4 or above, on *any* supported operating system (z/OS, UNIX, etc.) and hardware.
- Microsoft Excel 2002 or above (also referred to as Microsoft Excel XP).

SAMPLE DATA

The code in this paper uses the PHCVital SAS table, available in the download package of this paper. Figure 2 and Figure 3 show the column definitions and representative data values, respectively. The PROTOCOL column indicates whether the patient was enrolled in clinical trial "ABC 123" or "XYZ 987". The table is sorted by the PROTOCOL column, then PATIENT, and then VITTIME.



Column Name	Type	Length	Format	Informat	Label
PROTOCOL	Text	7			Protocol Identifier
PATIENT	Number	8			Patient Identifier
VITTIME	Number	8	6.2		Vital Signs: Time
VITRRATE	Number	8	2.		Respiratory Rate (/min)
VITHRATE	Number	8	3.		Heart Rate (bpm)
VITSYSBP	Number	8	3.		SBP (mmHg)
VITDIABP	Number	8	3.		DBP (mmHg)

Figure 2. Column Definitions for the PHCVital SAS Table

	Protocol Identifier	Patient Identifier	Respiratory Rate (/min)	Vital Signs: Time	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)
1	ABC 123	1	16	-2.00	105	135	84
2	ABC 123	1	9	-1.00	59	128	73
3	ABC 123	1	9	0.02	64	126	83
4	ABC 123	1	9	0.08	64	133	91
5	ABC 123	1	12	0.17	72	136	90
6	ABC 123	1	14	0.25	60	130	82
7	ABC 123	1	16	1.00	73	135	73
8	ABC 123	1	12	2.00	72	124	65
9	ABC 123	1	12	3.00	81	132	77
10	ABC 123	1	21	4.00	80	120	70
11	ABC 123	1	19	5.00	80	120	70
12	ABC 123	1	16	8.00	96	150	90
13	ABC 123	1	20	24.00	80	120	70
14	ABC 123	2	15	-2.00	72	94	56
15	ABC 123	2	12	-1.00	55	135	78
16	ABC 123	2	12	0.02	50	127	77
17	ABC 123	2	14	0.08	58	104	73
18	ABC 123	2	13	0.17	54	123	79
19	ABC 123	2	18	0.25	55	122	75
20	ABC 123	2	24	1.00	60	116	78
3240	XYZ 987	110	10	0.02	62	134	98
3241	XYZ 987	110	10	0.08	66	154	98
3242	XYZ 987	110	8	0.17	64	148	90
3243	XYZ 987	110	12	0.25	66	148	91
3244	XYZ 987	110	12	1.00	56	148	79
3245	XYZ 987	110	18	2.00	58	115	59
3246	XYZ 987	110	16	3.00	67	113	58
3247	XYZ 987	110	16	4.00	70	123	56
3248	XYZ 987	110	20	5.00	72	110	70
3249	XYZ 987	110	20	8.00	88	125	80
3250	XYZ 987	110	22	24.00	76	110	70

Figure 3. Representative Data Values for the PHCVital SAS Table

TECHNIQUE #1: USING THE EXPORT PROCEDURE TO CREATE CSV FILES

The first technique for getting SAS data into Excel uses the EXPORT procedure to read the data from a SAS table and write it to two CSV files. The procedure is available only for Windows and UNIX operating systems, and is limited to creating delimited files if only Base SAS is licensed. The SAS/ACCESS® Interface to PC Files software must be licensed to create native Microsoft Excel workbooks (SAS Institute Inc. 2012a, 2011a).

This code creates the two CSV files:

```
proc export data=sample.PHCVital (where=(protocol='ABC 123'))
  outfile='output-directory\VitalSigns-ExportABC.csv'
  dbms=csv label replace;
run; quit;

proc export data=sample.PHCVital (where=(protocol='XYZ 987'))
  outfile='output-directory\VitalSigns-ExportXYZ.csv'
  dbms=csv label replace;
run; quit;
```

The EXPORT procedure is run twice using the WHERE data set option because the WHERE and BY statements are not supported in that procedure. The DBMS and LABEL options specify that a CSV file is created that contains column labels when available, instead of column names. If the output file exists, it is overwritten because the REPLACE option is specified.

CSV is a plain-text format consisting of data fields separated by commas. Because the exported files only contain the data values (Figure 4), no Excel features such as formats, formulas, and colors, can be supported.

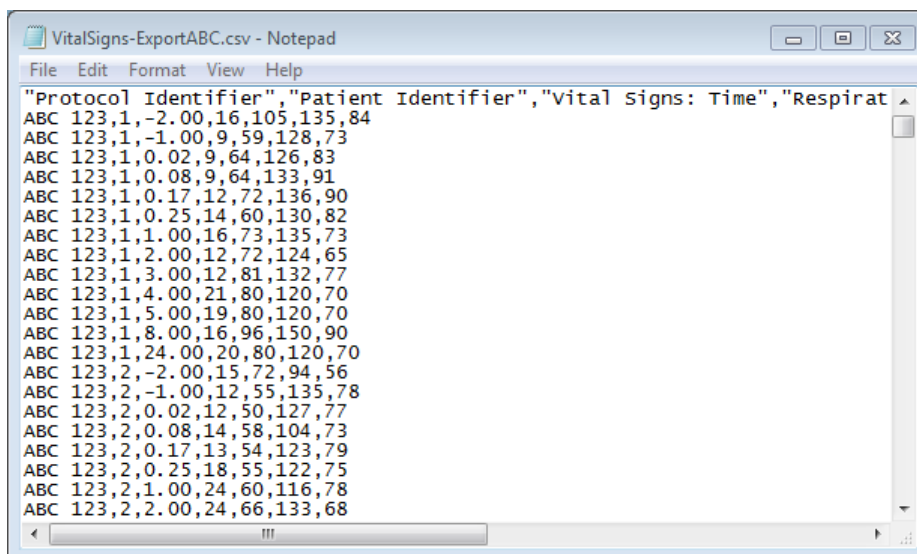


Figure 4. PROC EXPORT-Generated CSV File

Because Excel cannot automatically create a multi-sheet workbook from delimited files, we must manually import the data from each CSV file into its own worksheet.

Follow these steps to import the CSV files if you're using Excel 2010:

1. Select the **Data** tab, and then select **Get External Data ➔ From Text** in the **Get External Data** group.
2. In the "Import Text File" dialog, navigate to the directory that contains the CSV files, select **VitalSigns-ExportABC.csv**, and then click **Import** to start the "Text Import Wizard".
3. In Step 1 of the wizard, select **Delimited** in the **Original data type** group, and then click **Next >**.
4. Select **Comma** in the **Delimiters** group and then click **Next >**.
5. Click **Finish**, choose cell **A1** for the location to place the data, and then click **OK** to complete the import process.
6. Click an empty worksheet and repeat the steps for the "VitalSigns-ExportXYZ.csv" file.

If you're using Excel 2007 the steps are:

- Select the **Data** tab, and then select **From Text** in the **Get External Data** group.
- Continue with steps 2-6 above.

For Excel 2002-2003, use these steps:

- Select **Data** ➔ **Import External Data** ➔ **Import Data...**
- In the "Select Data Source" dialog, navigate to the directory containing the CSV files, select **VitalSigns-ExportABC.csv**, and then click **Open** to start the "Text Import Wizard".
- Continue with steps 3-6 above.

Although creating and importing CSV files can sometimes be useful, by comparing Figure 1 and Figure 5, you can see that the EXPORT procedure cannot be used to create the workbook that we want. Because CSV files are limited to data values, PROC EXPORT is limited to exporting only data values to Excel.

	A	B	C	D	E	F	G	H
1	Protocol Identifier	Patient Identifier	Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
2	ABC 123	1	-2	16	105	135	84	
3	ABC 123	1	-1	9	59	128	73	
4	ABC 123	1	0.02	9	64	126	83	
5	ABC 123	1	0.08	9	64	133	91	
6	ABC 123	1	0.17	12	72	136	90	
7	ABC 123	1	0.25	14	60	130	82	
8	ABC 123	1	1	16	73	135	73	
9	ABC 123	1	2	12	72	124	65	
10	ABC 123	1	3	12	81	132	77	
11	ABC 123	1	4	21	80	120	70	
12	ABC 123	1	5	19	80	120	70	
13	ABC 123	1	8	16	96	150	90	
14	ABC 123	1	24	20	80	120	70	
15	ABC 123	2	-2	15	72	94	56	
16	ABC 123	2	-1	12	55	135	78	
17	ABC 123	2	0.02	12	50	127	77	
18	ABC 123	2	0.08	14	58	104	73	
19	ABC 123	2	0.17	13	54	123	79	
20	ABC 123	2	0.25	18	55	122	75	
21	ABC 123	2	1	24	60	116	78	
22	ABC 123	2	2	24	66	133	68	
23	ABC 123	2	3	24	71	115	64	
24	ABC 123	2	4	23	76	130	70	
25	ABC 123	2	5	20	72	130	70	
26	ABC 123	2	8	24	80	120	80	
27	ABC 123	2	24	32	96	160	100	
28	ABC 123	3	-2	20	96	157	104	
29	ABC 123	3	-1	12	86	100	68	

Figure 5. Excel Workbook Generated Using PROC EXPORT and Manually Importing the CSV Files

To save a copy of the file in Excel binary (.xls) format using Excel 2002, 2003, or 2010, select **File** ➔ **Save As** and then, from the **Save as type** drop-down list, select **Microsoft Excel Workbook (*.xls)**. If you're using Excel 2007, click the **Microsoft Office Button**, and then select **Save As** ➔ **Excel 97-2003 Workbook**. If you're using Excel 2007 or 2010 and want to save the document in the Microsoft Office Open XML format, choose **Excel Workbook (*.xlsx)** from the **Save as type** drop-down list.

After the file is saved in a native Excel format, you can use Excel to add features such as formats, formulas, colors, or any other feature supported by Excel.

OUTPUT DELIVERY SYSTEM (ODS) BASICS

The Output Delivery System is capable of creating files that can be opened by Excel, and is used in the next two techniques. ODS is the part of Base SAS that enables you to generate different types of output from your procedure code. An *ODS destination* controls the type of output that is generated (HTML, RTF, PDF, etc.). An *ODS style* controls all aspects of the appearance of the output, and Base SAS ships with more than 50 different styles.

The general format for using ODS is:

```
ods destination-name file='file-name.ext' style=style-name ... ;
    * Your SAS procedure code here;
ods destination-name close;
```

The ODS statements needed to generate HTML output from your procedure code using the Printer style are:

```
❶ ods _all_ close;

❷ ods html file='file-name.htm' style=Printer;
    * Your SAS procedure code here;
❸ ods html close;
```

The first ODS statement (❶) closes all destinations that are open, because we want to generate only HTML output.

The second ODS statement (❷) uses the HTML destination to generate the HTML output and then store the output in a file. The STYLE option controls the appearance of the output, such as the font and color scheme. To see a list of ODS styles that are available for use at your site, submit the following SAS code:

```
ods _all_ close;
ods listing;
proc template; list styles; run; quit;
```

To find the SAS code that generates sample output for the ODS styles available on your system, click the **Full Code** tab in SAS Sample 36900 (SAS Institute Inc. 2009).

The third ODS statement (❸) closes the HTML destination and releases the HTML file so that it can be opened with a Web browser or another application.

You create other types of output by changing the destination name in the second (❷) and third (❸) ODS statements.

TECHNIQUE #2: USING THE ODS CSVALL DESTINATION TO CREATE CSV FILES

We can use the ODS CSVALL destination and the PRINT procedure to export the PHCVital table to CSV files "VitalSigns-ODSABC.csv" and "VitalSigns-ODSXYZ.csv". We use #BYVAL to display the current value of the PATIENT BY-variable in the document title, and the NOBYLINE system option prevents showing the same information in the document (SAS Institute Inc. 2011d).

```
title 'Subject ID: #BYVAL(patient)';
footnote;

options nobyline;

ods _all_ close;
```

```
ods csvall file='output-directory\VitalSigns-ODSABC.csv';
proc print data=sample.PHCVital noobs label;
  by protocol patient;
  var vittime vitrrate withrate vitsysbp vitdiabp;
  pageby patient;
  where (protocol eq 'ABC 123');
run; quit;
ods csvall close;

ods csvall file='output-directory\VitalSigns-ODSXYZ.csv';
proc print data=sample.PHCVital noobs label;
  by protocol patient;
  var vittime vitrrate withrate vitsysbp vitdiabp;
  pageby patient;
  where (protocol eq 'XYZ 987');
run; quit;
ods csvall close;

options byline;
```

Because CSV files contain only data values, the ODS STYLE option is not used. Figure 6 shows the result of manually importing the CSV files using the instructions in the *"Technique #1: Using the EXPORT Procedure to Create CSV Files"* section.

The output displayed in Figure 6 is similar to that of Figure 1, but lacks colors and formatting. If you want to save a copy of the file in a native Excel format (.xls or .xlsx), follow the instructions in the *"Technique #1: Using the EXPORT Procedure to Create CSV Files"* section. Then you can alter the file manually to add Excel-supported features.

The screenshot shows an Excel workbook with two sheets. Sheet1 contains data for Subject ID: 1, and Sheet2 contains data for Subject ID: 2. Both sheets have columns for Vital Signs: Time, Respiratory Rate (/min), Heart Rate (bpm), SBP (mmHg), and DBP (mmHg). The data is organized into rows for each subject, with time points ranging from -2 to 24 minutes.

Subject ID	Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)
Subject ID: 1	-2	16	105	135	84
	-1	9	59	128	73
	0.02	9	64	126	83
	0.08	9	64	133	91
	0.17	12	72	136	90
	0.25	14	60	130	82
	1	16	73	135	73
	2	12	72	124	65
	3	12	81	132	77
	4	21	80	120	70
Subject ID: 2	5	19	80	120	70
	8	16	96	150	90
	24	20	80	120	70
	-2	15	72	94	56
	-1	12	55	135	78
	0.02	12	50	127	77
	0.08	14	58	104	73
	0.17	13	54	123	79
	0.25	18	55	122	75
	1	24	60	116	78
Subject ID: 2	2	24	66	133	68
	3	24	71	115	64

Figure 6. Excel Workbook Generated Using ODS CSVALL and Manually Importing the CSV Files

TECHNIQUE #3: USING THE ODS EXCELPX TAGSET DESTINATION

While the two previous techniques produced CSV files, those files don't support the output and styling options that we want. To get those styling options we need to use the ExcelXP tagset, a type of ODS destination that creates ***XML output*** that can be opened with Excel. Unlike the delimited (CSV) files generated by the prior techniques, using the ODS ExcelXP tagset allows you to take advantage of rich Excel features, such as multiple worksheets, formats, and colors.

The Excel workbook in Figure 1 was created using the ExcelXP ODS tagset and the Printer ODS style supplied by SAS. The ExcelXP tagset creates an XML file that, when opened by Excel, is rendered as a multi-sheet workbook. All formatting and layout are performed by SAS; there is no need to hand-edit the Excel workbook. You simply use Excel to open the file created by ODS.

Because the ExcelXP ODS tagset creates files that conform to the Microsoft XML Spreadsheet Specification, you can create multi-sheet Excel workbooks that contain output from almost any SAS procedure. The exception is that the Microsoft XML Spreadsheet Specification does not support images, so the output from SAS/GRAPH® software procedures cannot be used (Microsoft Corporation 2001).

UPDATING THE EXCELPX TAGSET ON YOUR SYSTEM

The version of the ExcelXP tagset that is shipped with Base SAS 9 is periodically revised. There is currently no notification system for tagset updates. To ensure that you have a recent version, compare the ExcelXP tagset

version, displayed in the SAS log whenever the tagset is used, to the version available on the ODS Web site (SAS Institute Inc. [2013](#)).

Submit this code to display the tagset version number in the SAS log:

```
filename temp temp;

ods tagsets.ExcelXP file=temp;
ods tagsets.ExcelXP close;

filename temp clear;
```

If you're using a tagset that's more than 2 or 3 versions old, consider upgrading. See this author's earlier paper and SAS Usage Note 32394 for instructions (DelGobbo [2012](#), SAS Institute Inc. [2008b](#)).

BASIC SAS CODE TO CREATE THE EXCEL WORKBOOK

Here is the *basic* SAS code that creates the workbook in Figure 1:

```
title 'Subject ID: #BYVAL(patient)';
footnote;

ods _all_ close;

ods tagsets.ExcelXP file='output-directory\VitalSigns-ExcelXP.xml' style=Printer;
proc print data=sample.PHCVital noobs label;
  by protocol patient;
  var vittime vittrate vithrate vitsysbp vitdiabp;
  pageby patient;
run; quit;
ods tagsets.ExcelXP close;
```

The ExcelXP tagset generates the output, and the Printer style controls the appearance of the output. Unlike the previous techniques, the PRINT procedure is run only once to create a single file ("VitalSigns-ExcelXP.xml") with data from both trials. You should use the .xml extension instead of .xls or .xlsx, because Excel 2007 and 2010 display a warning if the .xml extension is not used (Microsoft Corporation [2013b](#)).

OPENING THE ODS EXCELXP OUTPUT WITH EXCEL

Follow these steps to open the "VitalSigns-ExcelXP.xml" file:

1. In Excel 2002, 2003, or 2010, select **File ➔ Open**
In Excel 2007, select **Office Button ➔ Open**.
2. Navigate to the "VitalSigns-ExcelXP.xml" file or type the path and file name in the **File name** field.
3. Click **Open**.

You can also navigate to the file using Microsoft Windows Explorer, and then double-click the file to open it with Excel.

Excel reads and converts the XML file to the Excel format. After the conversion, you can perform any Excel function on the data. Follow the instructions in the "*Technique #1: Using the EXPORT Procedure to Create CSV Files*" section if you want to save a copy of the file in a native Excel format (.xls or .xlsx).

The "VitalSigns-ExcelXP.xml" file displayed in Figure 7 resembles Figure 1, but has the following differences:

1. Each trial-patient combination appears in its own worksheet, resulting in 250 worksheets. Patient information should be grouped by trial (the PROTOCOL variable), resulting in only 2 worksheets.
2. Unattractive, default worksheet names are used.
3. Standard BY-group text (**Protocol Identifier=ABC 123 Patient Identifier=1**) precedes the table, instead of the TITLE statement text.
4. Not all of the data values in column **A** are displayed with two decimal places.

We can now change the basic SAS code to correct these problems. The complete SAS code used to create the workbook in Figure 1 is listed in the section *"The Final SAS Code"*.

Protocol Identifier=ABC 123 Patient Identifier=1					
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
-2	16	105	135	84	
-1	9	59	128	73	
0.02	9	64	126	83	
0.08	9	64	133	91	
0.17	12	72	136	90	
0.25	14	60	130	82	
1	16	73	135	73	
2	12	72	124	65	
3	12	81	132	77	
4	21	80	120	70	
5	19	80	120	70	
8	16	96	150	90	
24	20	80	120	70	

Figure 7. Initial Excel Workbook Generated Using the ODS ExcelXP Tagset

UNDERSTANDING AND USING THE EXCELXP TAGSET OPTIONS

The ExcelXP tagset supports many options that control both the appearance and functionality of the Excel workbook. Many of these tagset options are simply tied directly into existing Excel options or features. Tagset options are specified in an ODS statement using the **OPTIONS** keyword:

```
ods tagsets.ExcelXP options(option-name1='value1' option-name2='value2' ...) ... ;
```

Note that the value that you specify for a tagset option remains in effect until the ExcelXP destination is closed, or the option is set to another value. Because multiple ODS statements are allowed, it is good practice, in terms of functionality and code readability, to explicitly reset tagset options to their default value when you are finished using them.

For example:

```
ods tagsets.ExcelXP options(option-name='some-value');
* Some SAS procedure code here;
ods tagsets.ExcelXP options(option-name='default-value');
* Other SAS procedure code here;
```

NOTE: When specifying *additional* ODS statements, do not specify the FILE, STYLE, or any other keyword or option that is supported by ODS. Those options should be specified only on the initial ODS statement.

To see a listing of the supported options, submit this SAS code:

```
filename temp temp;

ods tagsets.ExcelXP file=temp options(doc='help');
ods tagsets.ExcelXP close;

filename temp clear;
```

The tagset information is printed to the SAS log. For your convenience, a listing of the supported options is included in the download package for this paper.

GROUPING OUTPUT BY TRIAL, AND USING BY-GROUP VALUES IN WORKSHEET NAMES

By default, the ExcelXP tagset creates a new worksheet when a SAS procedure creates new tabular output. Because a new table is created for each of the 250 combinations of PROTOCOL and PATIENT, the initial workbook contains 250 worksheets, each with a unique name as required by Excel.

We can fix this by setting the SHEET_LABEL option to specify the prefix to use for the worksheet names, and setting the value of the SHEET_INTERVAL option to *bygroup*. This combination generates 2 worksheets, one for each distinct value of the PROTOCOL BY-variable. Here is the additional ODS statement to handle the tagset options:

```
ods tagsets.ExcelXP file='output-directory\VitalSigns-ExcelXP.xml' style=Printer;

ods tagsets.ExcelXP options(sheet_interval='bygroup'
                             sheet_label='Trial');

proc print data=sample.PHCVital noobs label;
  by    protocol patient;
  var   vittime vitrrate vithrate vitsysbp vitdiabp;
  pageby patient;
run; quit;
ods tagsets.ExcelXP close;
```

Figure 8 displays the results of executing the code. If you do not want any text to prefix the BY-value, specify `sheet_label = ' '` (note the blank space between the quotation marks).

REPLACING THE BY-LINE TEXT WITH TITLE STATEMENT TEXT

BY-line text appears in the worksheets because the PRINT procedure is executed with a BY statement. Some of the text is redundant because the value of **Protocol Identifier** is displayed in the worksheet name. To omit the BY-line text, specify the SUPPRESS_BYLINES tagset option. Do not attempt to use the NOBYLINE *system* option, like we did with the CSVALL destination. Doing so would disable BY group processing in the ExcelXP tagset when SHEET_INTERVAL is set to *bygroup*.

By default, SAS titles and footnotes appear as Excel print headers and print footers, respectively, and are displayed when the Excel document is printed. To include title text on-screen, in the worksheet body, use the EMBEDDED_TITLES option:

```
ods tagsets.ExcelXP options(sheet_interval='bygroup'
                             sheet_label='Trial'
                             suppress_bylines='yes'
                             embedded_titles='yes');
```

Figure 9 displays a partial view of the workbook.

VitalSigns-ExcelXP.xml - Microsoft Excel

Protocol Identifier=ABC 123 Patient Identifier=1					
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
-2	16	105	135	84	
-1	9	59	128	73	
0.02	9	64	126	83	
0.08	9	64	133	91	
0.17	12	72	136	90	
0.25	14	60	130	82	
1	16	73	135	73	
2	12	72	124	65	
3	12	81	132	77	
4	21	80	120	70	
5	19	80	120	70	
8	16	96	150	90	
24	20	80	120	70	

Protocol Identifier=ABC 123 Patient Identifier=2					
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
-2	15	72	94	56	
-1	12	55	135	78	
0.02	12	50	127	77	
0.08	14	58	104	73	
0.17	13	54	123	79	
0.25	18	55	122	75	
1	24	60	116	78	
2	24	66	133	68	
3	24	71	115	64	

Trial ABC 123 Trial XYZ 987

Figure 8. ODS ExcelXP-Generated Workbook with Data Grouped by Trial, and BY-Values in Worksheet Names

VitalSigns-ExcelXP.xml - Microsoft Excel

<i>Subject ID: 1</i>					
Vital Signs: Time	Respiratory Rate (/min)	Heart Rate (bpm)	SBP (mmHg)	DBP (mmHg)	
-2	16	105	135	84	
-1	9	59	128	73	
0.02	9	64	126	83	
0.08	9	64	133	91	

Figure 9. ODS ExcelXP-Generated Workbook with TITLE Statement Text Replacing BY-Line Text

UNDERSTANDING AND USING ODS STYLE OVERRIDES

An ODS style contains *style elements*, each of which controls a particular part of the output. Style elements consist of collections of *style attributes*, such as the background color and font size.

You can alter the attributes used by specific parts of your SAS output by using *style overrides*. These specific parts of your SAS output are called *locations*. Figure 10 shows the locations that are pertinent to the PRINT procedure output (SAS Institute Inc. 2008a). The COLUMN location controls the appearance of data cells.

table		
obsheader	header	header
obs	column	column
obs	column	column
obs	column	column
obs	column	column
obs	column	column
bylabel	total	total
grandtotal	grandtotal	grandtotal
n		

Figure 10. Style Locations for the PRINT Procedure

Style overrides are supported by the PRINT, REPORT, and TABULATE procedures, and can be specified in several ways, the two most common formats being:

- ❶ `style(location)=[style-attribute-name1=value1 style-attribute-name2=value2 ...]`
- ❷ `style(location)=style-element-name`

The first format (❶) uses individual style attributes defined inline. For example, the following PROC PRINT code alters 3 attributes of the COLUMN location (data cells) for the MYVAR variable:

```
var myvar / style(column) = [background=yellow font_size=10pt just=left];
```

While this is the most commonly used format, it has some disadvantages. To use the same style override for different variables, you must apply it in multiple places, making your SAS code harder to read and maintain. And, if you want to use the style overrides in other SAS programs, you must copy the list of attribute name/value pairs to the new code. Style overrides of this type should be used sparingly.

The second format (❷) overcomes these problems by referencing a style element. Using this format involves creating a new style element, setting the style attributes within the element, and then using the *style element* name in your style override. This results in code that is easier to read, maintain, and re-use. Earlier papers by this author provide a detailed discussion of this topic (DeIGobbo 2008, 2009, 2010, 2011).

Refer to the ODS documentation for a full listing of style attributes (SAS Institute Inc. 2012b).

SPECIFYING AN EXCEL FORMAT FOR THE TIME VALUES

Our Excel workbook now closely resembles Figure 1, except that two decimal places are not always used for the time values. You could use SAS formats to control the Excel display values, but it is usually better to use Excel formats. This is because only the formatted values are transferred to Excel when SAS formats are used, and Excel may misinterpret the values in the absence of an Excel format. For example, the value 06708 is transferred to Excel when a ZIP Code is formatted with the SAS Z5. format, but the value is displayed as 6708 if an Excel format is not used.

Figure 11 shows the general structure of Excel number formats (Microsoft Corporation 2013a). The pound sign (#) in an Excel format is used to represent a numeric digit, excluding insignificant zeros, and a zero (0) displays a numeric digit, including insignificant zeros. Use zeros in Excel formats when you want to retain leading or trailing zeros.

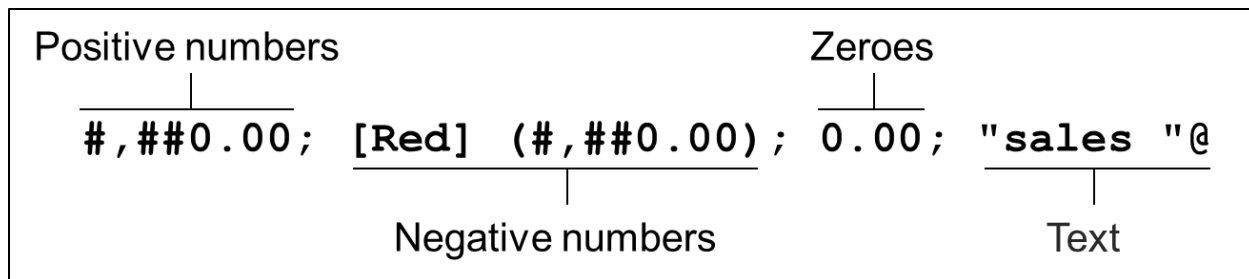


Figure 11. Structure of Excel Number Formats

Table 1 shows the results of applying the Excel format shown in Figure 11.

Raw Value	Formatted Value	Comment
.5	0.50	Leading & Trailing Zeros
5	5.00	Leading & Trailing Zeros
123	123.00	Trailing Zeros
1234	1,234.00	Trailing Zeros, Thousands Separator
-1234	(1,234.00)	Leading & Trailing Zeros, Thousands Separator, Red ()
0	0.00	Special Zero Handling
data	sales data	Special Text Handling

Table 1. Results of Applying Excel Format Shown in Figure 11

Because negative values do not need to be considered differently than positive values, we use the Excel format 0.00 with the ODS TAGATTR attribute to specify an Excel format. Be sure to quote the entire attribute value and include the `format:` keyword. To apply the style override to only the VITTIME column, we split the single VAR statement into two separate statements:

```
proc print data=sample.PHCVital noobs label;
  by protocol patient;
  var vittime / style(column) = [tagattr='format:0.00'];
  var vittrate vittrate vitsysbp vitdiabp;
  pageby patient;
run; quit;
```

With all of the code modifications in place, the resulting workbook matches the output shown in Figure 1.

THE FINAL SAS CODE

The final SAS code to create the Figure 1 output follows:

```
title 'Subject ID: #BYVAL(patient)';
footnote;

ods _all_ close;

ods tagsets.ExcelXP file='output-directory\VitalSigns-ExcelXP.xml' style=Printer;

ods tagsets.ExcelXP options(sheet_interval='bygroup'
                             sheet_label='Trial'
                             suppress_bylines='yes'
                             embedded_titles='yes');

proc print data=sample.PHCVital noobs label;
  by protocol patient;
  var vittime / style(column) = [tagattr='format:0.00'];
  var vitrrate vithrate vitsysbp vitdiabp;
  pageby patient;
run; quit;

ods tagsets.ExcelXP close;
```

See this author's earlier paper for more introductory information about using the ExcelXP tagset (DelGobbo [2012](#)).

DRIVING DATA TO EXCEL USING SAS SERVER TECHNOLOGY

You can incorporate dynamically-generated SAS output into Excel using the Application Dispatcher or the SAS® Stored Process Server. The Application Dispatcher is part of SAS/IntrNet® software. The SAS Stored Process Server is available starting with SAS 9 as part of SAS® Integration Technologies, and is included with server offerings that leverage the SAS Business Analytics infrastructure (for example, SAS® BI Server and SAS® Enterprise BI Server).

These products enable you to execute SAS programs from a Web browser or any other client that can open an HTTP connection to the Application Dispatcher or the SAS Stored Process Server. Both of these products can run on any platform where SAS is licensed. SAS does not need to be installed on the client machine.

The SAS programs that you execute from the browser can contain any combination of DATA step, procedure, macro, or SCL code. Thus, all of the code that has been shown up to this point can be executed by both the Application Dispatcher and the SAS Stored Process Server.

Program execution is typically initiated by accessing a URL that points to the SAS server program. Parameters are passed to the program as name/value pairs in the URL. The SAS server takes these name/value pairs and constructs SAS macro variables that are available to the SAS program.

Figure 12 shows a Web page that can deliver SAS output directly to Excel, using a Web browser as the client.

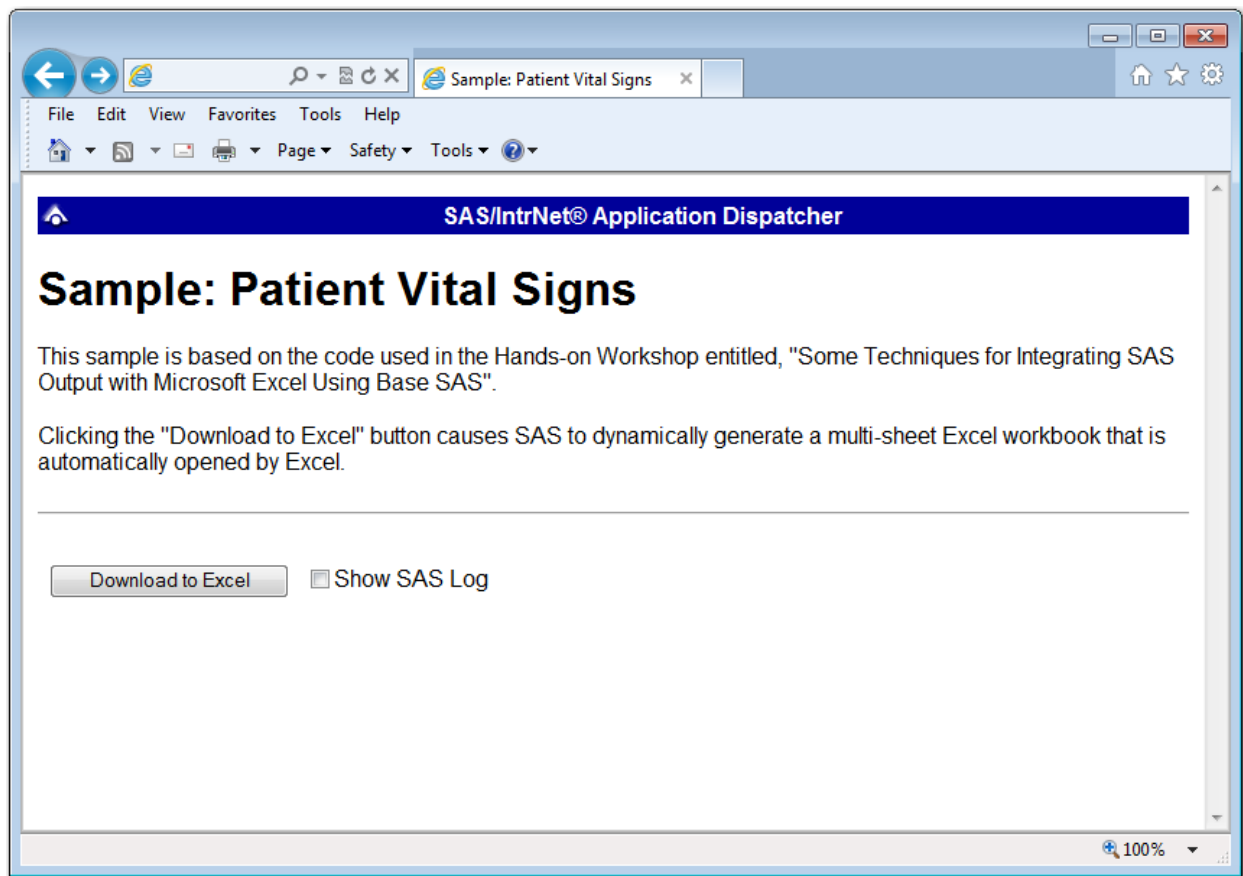


Figure 12. Web Page to Drive a SAS/IntrNet Application

Clicking **Download to Excel** executes a slightly modified version of the SAS code that we have been working on:

```
%let RV=%sysfunc(appsrv_header(Content-type,application/vnd.ms-excel));
%let RV=%sysfunc(appsrv_header(Content-disposition,attachment; filename=
"VitalSigns.xml")); * Ignore line wrapping;

ods listing close;
ods tagsets.ExcelXP file=_webout style=Printer;
* Remainder of the "final" SAS code;
ods tagsets.ExcelXP close;
```

The first APPSRV_HEADER function sets a MIME header that causes the SAS output to be opened by Excel, instead of being rendered by the Web browser. This statement is required.

The second APPSRV_HEADER function sets a MIME header that causes the file name to be displayed in the "File Download" dialog box. As you can see in Figure 13, the file name appears as **VitalSigns.xml**. This header might cause problems with some versions of Excel, so be sure to test your applications before deploying them in a production environment. This statement is optional.



Figure 13. File Download Dialog Box

The reserved fileref _WEBOUT is automatically defined by the SAS server and is always used to direct output from the SAS server to the client. Modify your existing ODS statement to direct the output to this fileref instead of to an external disk file.

When you click **Download to Excel** on the Web page and are presented with the "File Download" dialog box (Figure 13), you can click **Open** to immediately open your SAS output using Excel, or click **Save** to save a copy for later use.

This is just one example of how you can dynamically deliver SAS output to Excel. For more detailed information and other examples, see the SAS/IntrNet Application Dispatcher and SAS Stored Process documentation (SAS Institute Inc. 2011b, 2011c), as well as this author's earlier papers (DelGobbo 2002, 2003, 2004).

CONCLUSION

Table 2 lists the features that are supported by the 3 techniques:

Feature	PROC EXPORT	ODS CSVALL	ODS ExcelXP Tagset
Supported on all operating systems	● ¹	●	●
Multiple worksheets	● ²	● ²	●
BY value in worksheet name	● ²	● ²	●
Suppress BY line text	●	●	●
Variable labels in column headings	●	●	●
Support title, footnote & BY text		●	●
Title text contains subject ID		●	●
Multiple tables in a worksheet		●	●
Formatted column headings			●
Support for Excel number formats			●
Style formatting support			●
Little or no manual intervention			●

Table 2. Features Supported by PROC EXPORT and the ODS CSVALL and ExcelXP Tagset Destinations

¹Except z/OS ²Requires manual steps

The EXPORT procedure creates delimited files that are useful for exporting only data values to Excel - style formatting is not supported. The ODS CSVALL destination also creates delimited files, but supports some additional features, such as title, BY-line, and footnote text.

The SAS 9 ExcelXP ODS tagset provides an easy way to export your SAS data to Excel workbooks that contain multiple worksheets. By using ODS styles, style overrides, and a tagset that complies with the Microsoft XML Spreadsheet Specification, you can customize the output to achieve your design goals.

APPENDIX

VISUAL BASIC CODE TO CONVERT XML TO NATIVE EXCEL FORMATS

The author is developing a Visual Basic script that converts ExcelXP-generated files to native Excel .xls or .xlsx formats. Contact the author if you would like a copy of this experimental code.

REFERENCES

DelGobbo, Vincent. 2002. "Techniques for SAS® Enabling Microsoft® Office in a Cross-Platform Environment". *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. Available at <http://www2.sas.com/proceedings/sugi27/p174-27.pdf>.

DelGobbo, Vincent. 2003. "A Beginner's Guide to Incorporating SAS® Output in Microsoft® Office Applications". *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. Available at <http://www2.sas.com/proceedings/sugi28/052-28.pdf>.

DelGobbo, Vincent. 2004. "From SAS® to Excel via XML". Available at <http://support.sas.com/rnd/papers/sugi29/ExcelXML.pdf>.

DelGobbo, Vincent. 2008. "Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. Available at <http://www2.sas.com/proceedings/forum2008/192-2008.pdf>.

DelGobbo, Vincent. 2009. "More Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings09/152-2009.pdf>.

DelGobbo, Vincent. 2010. "Traffic Lighting Your Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings10/153-2010.pdf>.

DelGobbo, Vincent. 2011. "Creating Stylish Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings11/170-2011.pdf>.

DelGobbo, Vincent. 2012. "An Introduction to Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". *Proceedings of the SAS Global Forum 2012 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings12/150-2012.pdf>.

Microsoft Corporation. 2001. "XML Spreadsheet Reference". Available at [http://msdn2.microsoft.com/en-us/library/aa140066\(office.10\).aspx](http://msdn2.microsoft.com/en-us/library/aa140066(office.10).aspx).

Microsoft Corporation. 2013a. "Create or delete a custom number format". Available at <http://office.microsoft.com/assistance/hfws.aspx?AssetID=HP051995001033>.

Microsoft Corporation. 2013b. "When you open a file in Excel 2007, you receive a warning that the file format differs from the format that the file name extension specifies". Available at <http://support.microsoft.com/kb/948615>.

SAS Institute Inc. 2008a. "SAS® 9 Reporting Procedure Styles Tip Sheet". Available at <http://support.sas.com/rnd/base/ods/scratch/reporting-styles-tips.pdf>.

SAS Institute Inc. 2008b. "Usage Note 32394: Installing and Storing Updated Tagsets for ODS MARKUP". Available at <http://support.sas.com/kb/32/394.html>.

SAS Institute Inc. 2009. "Sample 36900: Instructions for viewing all of the style templates that are shipped with the SAS® System". Available at <http://support.sas.com/kb/36/900.html>.

SAS Institute Inc. 2011a. *SAS/ACCESS® 9.3 Interface to PC Files: Reference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/cdl/en/acpcref/63181/HTML/default/viewer.htm#titlepage.htm>

SAS Institute Inc. 2011b. *SAS/IntrNet® 9.3: Application Dispatcher*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/cdl/en/dispatch/62562/HTML/default/viewer.htm#p06h82ux8glu1pn16k9dxw8tjpyz.htm>

SAS Institute Inc. 2011c. *SAS® 9.3 Stored Processes: Developer's Guide*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/cdl/en/stpug/62758/HTML/default/viewer.htm#titlepage.htm>

SAS Institute Inc. 2011d. "TITLE Statement". *SAS® 9.3 Statements: Reference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/cdl/en/lestmtsref/63323/HTML/default/viewer.htm#p10gcmrmf83iaxn1ilrx4pra969n.htm>

SAS Institute Inc. 2012a. "Export Procedure". *Base SAS® 9.3 Procedures Guide, Second Edition*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm#n045uxf7ll2p5on1ly4at3vdpd47e.htm>

SAS Institute Inc. 2012b. SAS® 9.3 *Output Delivery System User: User's Guide*. Cary, NC: SAS Institute Inc.
Available at <http://support.sas.com/documentation/cdl/en/odsug/62755/HTML/default/viewer.htm#p15bfqggegial0n1j3ze57yaqljr.htm>.

SAS Institute Inc. 2013. "ODS MARKUP Resources". Available at <http://support.sas.com/rnd/base/topics/odsmarkup/>.

ACKNOWLEDGEMENTS

The author would like to thank Chris Barrett of SAS Institute Inc. for his valuable contributions to this paper.

RECOMMENDED READING

DelGobbo, Vincent. 2013. "Vince DelGobbo's ExcelXP Tagset Paper Index".
Available at <http://www.sas.com/events/cm/867226/ExcelXPPaperIndex.pdf>.

SAS Institute Inc. 2013. "Quick Reference for the TAGSETS.EXCELXP Tagset".
Available at http://support.sas.com/rnd/base/ods/odsmarkup/excelxp_help.html

SAS Institute Inc. 2010. "An Introduction to Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®". Available at <http://www.sas.com/reg/gen/corp/867226?page=Resources>

CONTACT INFORMATION

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

Vincent DelGobbo
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
sasvcd@SAS.com



<http://www.sas.com/reg/gen/corp/867226?page=Resources>

If your registered in-house or local SAS users group would like to request this presentation as your annual SAS presentation (as a seminar, talk, or workshop) at an upcoming meeting, please submit an online User Group Request Form (from support.sas.com/sasusersupport/usergroups/support) at least eight weeks in advance.

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.