Paper HW05-2011

Creating Stylish Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®

Vincent DelGobbo, SAS Institute Inc., Cary, NC

ABSTRACT

This paper explains how to use Base SAS® 9 software to create multi-sheet Excel workbooks (for Excel versions 2002 and later). You learn step-by-step techniques for quickly and easily creating attractive multi-sheet Excel workbooks that contain your SAS output using the ExcelXP ODS tagset and ODS styles. The techniques that are presented in this paper can be used regardless of the platform on which SAS software 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 provides you with step-by-step instructions for using Base SAS 9.1 or later to create the Excel workbook shown in Figure 1.

Microsoft Excel - ProstateCancer.xml

File Edit View Insert Format Tools Data SAS Window Help

	A	B	C	D	E	F	G	H	I
1	Subject ID	Age in Years	Size of Primary Tumor (cm2)	Status	History of Cardiovascular Disease	EKG Outcome	Bone Metastases	Systolic BP	Diastolic BP
2	5	67	34	alive		normal		170	100
3	7	75	13	dead - heart or vascular		benign		140	100
4	8	73	3	alive	Yes	heart strain		170	110
5	14	55	4	dead - prostatic ca	Yes	heart strain		160	90
6	17	64	6	alive		normal		140	80
7	23	61	8	alive		normal		110	80
8	27	76	12	alive		old MI		120	80
9	28	71	11	dead - prostatic ca		normal		120	70
10	34	79	26	dead - prostatic ca		heart strain		160	90
11	37	72	2	dead - other specific non-ca		heart strain		160	100
12	41	72	12	alive	Yes	heart strain		170	80
13	44	74	26	alive		heart block or conduction def		160	80
14	50	74	20	alive	Yes	old MI		110	60
15	51	73	18	dead - prostatic ca		normal		160	70
16	55	78	24	dead - prostatic ca	Yes	old MI		180	80
17	60	60	7	dead - other ca		normal		120	90
18	62	77	24	dead - cerebrovascular		old MI		210	90
19	64	70	26	dead - prostatic ca		normal	Yes	120	80
20	69	80	34	dead - other specific non-ca		normal	Yes	160	90
21	73	70	24	alive		benign		150	70
22	77	72	5	dead - heart or vascular	Yes	old MI		220	130
23	78	69	4	dead - other specific non-ca	Yes	normal		130	70
24	85	72	9	dead - respiratory disease	Yes	normal		170	110
25	91	84	8	dead - respiratory disease	Yes	benign		150	100
26	96	77	7	alive		normal		140	70
27	98	76	3	alive	Yes	old MI		120	70
28	104	76	3	alive	Yes	normal		160	90
29	109	77	0	alive		benign		100	60
30	112	73	61	dead - prostatic ca	Yes	heart strain		130	80
31	116	74	1	dead - heart or vascular	Yes	old MI		130	80

Placebo | 0.2 mg Estrogen | 1.0 mg Estrogen | 5.0 mg Estrogen

Ready NUM

Figure 1. First Worksheet of Multi-Sheet Excel Workbook Generated by the ExcelXP ODS Tagset

The workbook contains four worksheets containing randomized clinical trial data comparing four treatments for patients with prostate cancer (Andrews 1985). Different background colors are applied to alternating rows to make them easier to read, and patients who died due to cardiovascular disease, a possible side effect of the treatment, are highlighted in orange (Byar 1980; Bailar 1970).

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 "Creating Stylish Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®".

The code in this paper was tested using SAS® 9.1.3 Service Pack 4, and Microsoft Excel 2003 Service Pack 2 software.

REQUIREMENTS

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

- Base SAS 9.1.3 or later, on *any* supported operating system (z/OS, UNIX, etc.) and hardware.
- Microsoft Excel 2002 or later (also referred to as Microsoft Excel XP).
- **An updated version of the SAS ExcelXP ODS tagset.**
  For information about obtaining an updated version of the tagset, see "The ExcelXP Tagset" section later in this paper.

LIMITATIONS

Because the ExcelXP ODS tagset creates files that conform to the Microsoft XML Spreadsheet Specification, you can create multi-sheet Excel workbooks containing the 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).

You can use ExcelXP tagset options with all procedure output, but ODS style overrides apply only to the PRINT, REPORT, and TABULATE procedures. Tagsets and style overrides are discussed in the sections "Working with the ExcelXP Tagset Options" and "Using the XLsansPrinter Style and Style Overrides", respectively.

SAMPLE DATA

Table 1 presents abbreviated information about the SAS table "ProstateCancer" that is used to create the Excel workbook shown in Figure 1. An asterisk (*) is used as a split character in some variable labels to control text wrapping in the column headings.

Variable Name	Variable Label	Variable Type	Typical Values
RX	Drug	Numeric	1, 2, 3, or 4
PatNo	Subject*ID	Numeric	1 – 506
Age	Age in*Years	Numeric	48 – 89
SZ	Size*of*Primary Tumor*(cm2)	Numeric	0 – 69
Status	Status	Character	alive, dead – prostatic ca
HX	History of*Cardiovascular*Disease	Numeric	0 or 1
EKG	EKG Outcome	Character	normal, heart strain
BM	Bone*Metastases	Numeric	0 or 1
SBP	Systolic BP	Numeric	80 – 300
DBP	Diastolic BP	Numeric	40 – 180

Table 1. Representative Data Values in the SAS Table "ProstateCancer"

The REPORT procedure is run against this data to create the workbook. The worksheet names correspond to the formatted values of the BY variable "RX".

OUTPUT DELIVERY SYSTEM (ODS) BASICS

ODS is the part of Base SAS software 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 the appearance of the output. In this paper, we use a type of ODS destination, called a *tagset*, that creates XML output that can be opened with Excel. This tagset, named ExcelXP, creates an Excel workbook that has multiple worksheets.

The Excel workbook in Figure 1 was created using the ExcelXP ODS tagset and the user-defined "XLsansPrinter" ODS style. 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.

Here are the general ODS statements needed to generate XML output that is compatible with versions 2002 and later of Excel:

```
❶ ods listing close;

❷ ods tagsets.ExcelXP file='file-name.xml' style=style-name ... ;
    *  Your SAS procedure code here;
❸ ods tagsets.ExcelXP close;
```

The first ODS statement (❶) closes the LISTING destination, which writes output to either a listing file in batch mode, or to the Output window when SAS is run interactively. Because we want to generate only XML output for use with Excel, we close the LISTING destination.

The second ODS statement (❷) uses the ExcelXP tagset to generate the XML output and then store the output in a file. 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 2011). 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 listing;
proc template; list styles; run; quit;
```

SAS code that generates sample output for the ODS styles available on your system can be found by clicking the "Full Code" tab in SAS Sample 36900 (SAS Institute Inc. 2009a).

The third ODS statement (❸) closes and releases the XML file so that it can be opened with Excel.

Although you can store your output on a local disk (where SAS software is installed), or on a network-accessible disk, here are some good reasons to store your SAS output on a Web server:

- The files are available to anyone who has network access.
- The XML files can be accessed by Web-enabled applications other than Excel.
- You can take advantage of Web server authentication and security models.

Note: If you place the files where users can access them over a network, you should set file permissions to prevent accidental alteration.

OPENING ODS OUTPUT WITH EXCEL

To open an ODS-generated file that is stored on a Web server, follow these steps:

1. In Excel 2002, 2003 or 2010, select **File ➨ Open**
   In Excel 2007 select **Office Button ➨ Open**.
2. In the **File name** field, specify the full URL for the file that you want to open. For example, **http://*Web-server/directory/file-name*.xml**.
3. Click **Open** to import the XML file.

To open ODS-generated files from a local or network-accessible disk, follow the same steps, except in step 2 you should either navigate to the file or type the path and file name in the **File name** field. You can also navigate to the file using Microsoft Windows Explorer, and then double-click the file to open it with Excel.

If you encounter a "problem during load" error when you attempt to open the file with Excel, see the appendix in the section "Diagnosing Excel Load Errors".

Excel reads and converts the XML file to the Excel format. After the conversion, you can perform any Excel function on the data. 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**.

SETTING UP THE ODS ENVIRONMENT

Our sample code employs a user-defined style named "XLsansPrinter" and an updated version of the ExcelXP tagset. The following statements define the location where the style and tagset are stored on your system:

```
❶ libname mylib 'some-directory'; * Location to store tagsets and styles;

❷ ods path mylib.tmplmst(update) sashelp.tmplmst(read);
```

The LIBNAME statement (❶) specifies where to store the user-defined tagsets and styles. Although you can temporarily store tagsets and styles in the WORK library, it is more efficient to create them once, and then store them in a permanent library so that you can reference them in other SAS programs.

The ODS PATH statement (❷) specifies the locations of, and the order in which to search for, ODS tagsets and styles. Notice that the access mode for `mylib.tmplmst` is specified as "update" and the access mode for `sashelp.tmplmst` is specified as "read". Because ODS searches the path in the order given, and the access mode for `mylib.tmplmst` is "update", PROC TEMPLATE, used later in this paper, creates and stores tagsets and styles in a file named "tmplmst.sas7bitm" in the directory that is associated with the "mylib" library.

THE EXCELXP TAGSET

Once you have issued the appropriate ODS PATH statement, you can import an updated version of the ExcelXP tagset and use it in your SAS programs. The version of the tagset used in this paper can be found in the download package on the SAS Presents Web site at support.sas.com/saspresents. Find the entry "Creating Stylish Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS". The download package contains a file named "ExcelXP.sas", which contains the SAS code for creating the ExcelXP tagset. Save a local copy of this file, and then submit the following SAS code to make the tagset available:

```
%include 'ExcelXP.sas';  * Specify the path to the file, if necessary;
```

You need to submit this code only once. The ExcelXP tagset is imported and stored in the directory corresponding to the "mylib" library. All of your future SAS programs can access the tagset by specifying the correct LIBNAME and ODS PATH statements. (See "Setting up the ODS Environment".)

The ExcelXP tagset supports many options that control both the appearance and functionality of the Excel workbook. To see a listing of the supported options, submit the following SAS code:

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

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.

IMPORTANT NOTE

The version of the ExcelXP tagset that was shipped with Base SAS®9 has undergone many revisions since its initial release. To take advantage of the features discussed in this paper, you must download an updated version of the tagset and install it on your system as described previously. The version of the tagset used in this paper can be found in the download package on the SAS Presents Web site, as noted above. A recent version of the tagset is available from the ODS Web site (SAS Institute Inc. 2011).

A BRIEF ANATOMY OF ODS STYLES

ODS styles control all aspects of the appearance of the output, and Base SAS software ships with over 40 different styles. A style contains *style elements,* each of which controls a particular part of the output. For example, a style element named "header" controls the appearance of column headings. Style elements consist of collections of *style attributes*, such as the background color and font size.

Use the ODS tagset named "style_popup" when you need to determine the attributes of style elements. The tagset creates an HTML file that, when viewed using the SAS Results window or the Microsoft Internet Explorer Web browser, displays style element information in popup windows (SAS Institute Inc. 2009d).

For example, to see the style attributes of the "header" style element of the "sansPrinter" style, follow these steps:

1. Submit this SAS code:

```
ods results;
ods tagsets.style_popup path='output-directory' file='temp.htm'
  style=sansPrinter;

 *  Your SAS procedure code here;

ods tagsets.style_popup close;
```

2. View the HTML output using the SAS Results window or Microsoft Internet Explorer.

3. Click on a column heading to display a popup window containing the style element name ("header") and style attribute information for that cell (Figure 2).

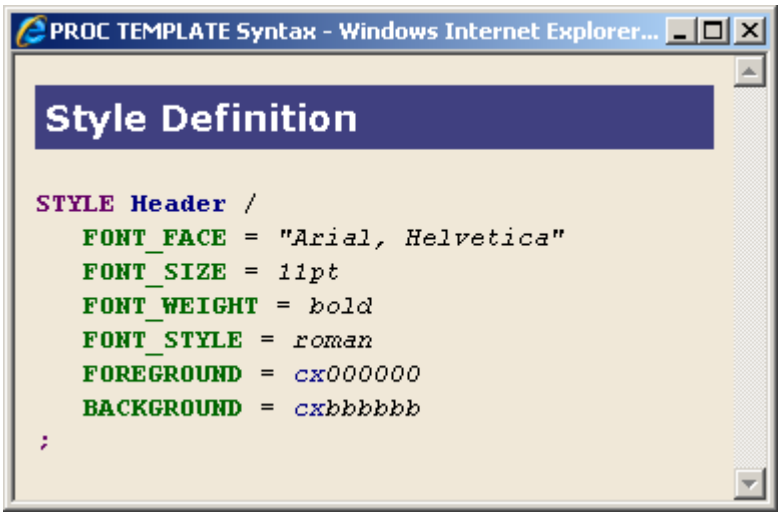


Figure 2. Style Attributes of the "header" Style Element

This style element contains the style attributes "font_face", "font_size", and so on. The next section describes using the TEMPLATE procedure to create a new style by first copying an existing style, and then modifying the copy.

CREATING YOUR OWN STYLE: THE XLSANSPRINTER STYLE

The workbook shown in Figure 1 was created with a user-defined ODS style named "XLsansPrinter", which is based on the "sansPrinter" style that is supplied by SAS. The "XL" in the "XLsansPrinter" name indicates that this style is intended for use with Excel.

The "XLsansPrinter" style was created by using the TEMPLATE procedure to copy the "sansPrinter" style, and then change the existing "header" style element supplied by SAS. The complete code for creating the "XLsansPrinter" style can be found in the appendix of this paper, "Code for Creating the XLsansPrinter Style".

Although you can use the TEMPLATE procedure to create a new style without copying an existing style, it is usually easier to copy an existing style that is close to what you want, and then make modifications to the copy. The code below creates the user-defined "XLsansPrinter" style by copying the "sansPrinter" ODS style supplied by SAS:

```
proc template;
  define style styles.XLsansPrinter;
    parent = styles.sansPrinter;
  end;
run; quit;
```

Because this TEMPLATE procedure code does not contain statements that change any style elements, the "XLsansPrinter" style is an exact copy of the "sansPrinter" style. That is, the "XLsansPrinter" style inherits all of the style elements and attributes of the parent style, "sansPrinter".

Style elements are created using the STYLE statement, which follows this general format:

```
style new-style-element-name from existing-style-element-name /
  style-attribute-specifications;
```

The following code, when added to the initial definition of the "XLsansPrinter style" (above), illustrates how you can change the value of an existing style attribute, and also add new attributes.

```
        ❶          ❷
style header from header /
  font_size = 10pt
  just      = center
  vjust     = bottom;
```

The first occurrence of "header" (❶) indicates the name of the style element being created. The second occurrence (❷) specifies the name of an existing style element to use as the parent. New style elements inherit all of the style attributes of the parent element. The code above creates a style element named "header", which inherits all of the style attributes of the existing style element named "header" shown in Figure 2. Then the font size is changed from "11pt" to "10pt", and new style attributes are added to control the horizontal and vertical justification. All other style attributes are inherited from the existing "header" style element, and they remain unchanged. Figure 3 is a graphic representation of the code above.

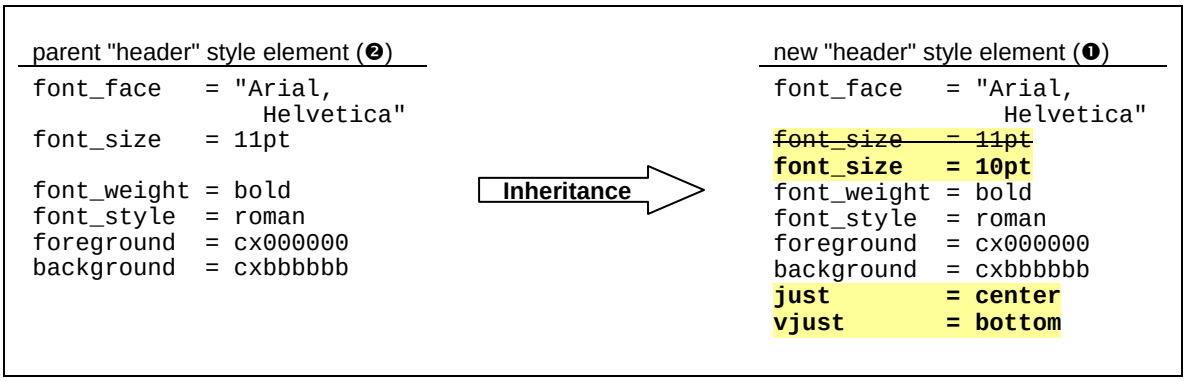


Figure 3. Creating the "header" Style Element

When you run the TEMPLATE procedure code provided in the appendix, you generate the "XLsansPrinter" style containing the modified version of the "header" style element.

At this point you should run the TEMPLATE procedure code found in the appendix of this paper to create the "XLsansPrinter" style on your system. You need to perform this step only once because the "XLsansPrinter" style is stored in the directory corresponding to the MYLIB library, and all of your future SAS programs can access the style by adding the correct LIBNAME and ODS PATH statements. (See the "Setting Up the ODS Environment" section.)

Because an in-depth discussion of creating and using ODS styles is beyond the scope of this paper, see the chapter about the TEMPLATE procedure in the ODS documentation (SAS Institute Inc. 2009b).

USING ODS TO CREATE THE MULTI-SHEET EXCEL WORKBOOK

By default, the ExcelXP tagset creates a new worksheet each time a SAS procedure creates new tabular output. The REPORT procedure creates four worksheets, one for each distinct value of the BY variable "RX".

SAS CODE TO CREATE THE EXCEL WORKBOOK

Here is a listing of the *basic* SAS code used to create the Excel workbook.

```
*;
*  Create formats to make drug codes
*  and Boolean values more user-friendly
*;

❶ proc format;

    value rx 1 = 'Placebo'
```

```
              2 = '0.2 mg Estrogen'
              3 = '1.0 mg Estrogen'
              4 = '5.0 mg Estrogen';

    value boolean 0 = ' '
                  1 = 'Yes';
  run; quit;

  ods listing close;

❷ ods tagsets.ExcelXP path='output-directory' file='ProstateCancer.xml'
    style=XLsansPrinter;

  title;
  footnote;

  *  One worksheet created for each distinct BY value;

❸ proc report data=sample.ProstateCancer nowindows split='*';
    by RX;

    column PatNo Age SZ Status HX EKG BM SBP DBP;

    define PatNo  / display;
    define Age    / display;
    define SZ     / display;
    define Status / display;
❹   define HX     / display format=boolean.;
    define EKG    / display;
    define BM     / display format=boolean.;
    define SBP    / display;
    define DBP    / display;

❺   compute PatNo;
      * Placeholder for row highlighting;
    endcomp;

❻   compute Status;
      * Placeholder for cell highlighting;
    endcomp;

❼   format RX rx.;

  run; quit;

  ods tagsets.ExcelXP close;
```

The FORMAT procedure (❶) is run to create formats that display more useful values for the "RX", "HX", and "BM" variables. Table 1 lists the raw data values of these variables.

As you can see in the ODS statement (❷), the ExcelXP tagset generates the output, and the "XLsansPrinter" style controls the appearance of the output. PROC REPORT (❸) is run with a BY statement and creates four worksheets, one for each distinct value of the variable "RX". The user-defined formats "boolean" and "rx" are applied to the "HX", "BM", and "RX" variables (❹ and ❼).

The first COMPUTE block (❺) is a placeholder for code that changes the background color of alternating rows of data. The second block (❻) is a placeholder for "traffic lighting" the "Status" variable data. Traffic lighting is the process of applying formatting to data in order to draw attention to specific values.

Figure 4 displays the results of executing the basic SAS code, and opening the resulting "ProstateCancer.xml" file with Excel. Notice that Figure 4 does not match Figure 1. The following problems are exhibited in Figure 4:

1. Alternating rows do not have a blue background.
2. None of the values in any of the worksheets are traffic-lighted.
3. Data values in all columns except "Status" and "EKG" should be centered.
4. Just the formatted value of the BY group variable "RX" should be used for the worksheet name (without a prefix).
5. Standard BY group text ("Drug=Placebo") precedes the table, and it is redundant with the worksheet name.
6. Columns B through G do not contain Excel AutoFilters.
7. Some of the columns are too narrow, causing data values to be obscured.

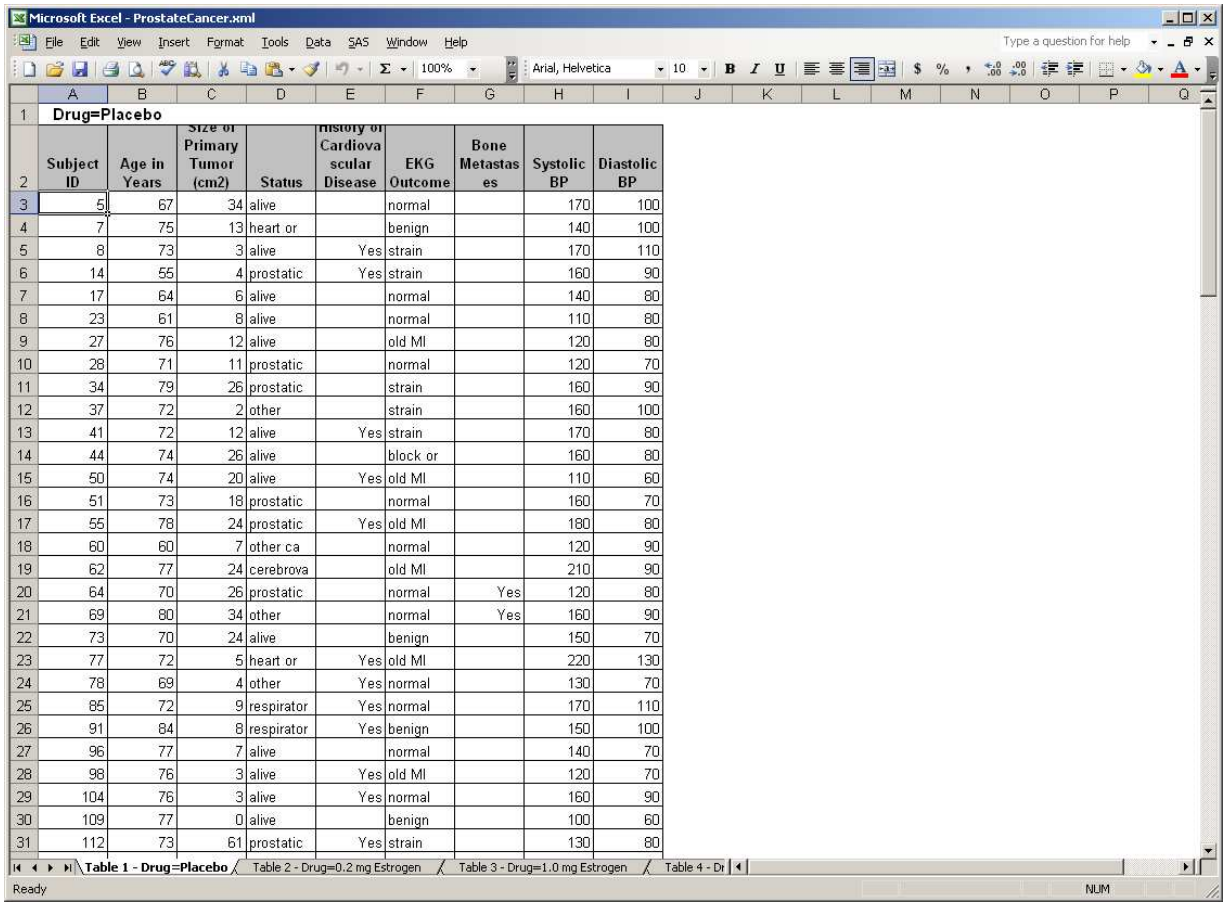


Figure 4. Initial ExcelXP ODS Tagset-Generated Workbook

We can now change the basic SAS code to correct these problems.

USING THE XLSANSPRINTER STYLE AND STYLE OVERRIDES

As mentioned earlier, the user-defined "XLsansPrinter" style is very similar to the ODS "sansPrinter" style supplied by SAS, but the two styles do differ. The column headings of the "XLsansPrinter" style use a font that is centered 10-point instead of left-justified 11-point. Also, the vertical justification of text in column headings is set to "bottom". These differences are a result of the modifications made to the "header" style element which is supplied by SAS, as shown in the code in the appendix.

Understanding and Using ODS Style Overrides

You can alter the attributes or style elements used by specific parts of your SAS output by using style overrides. These specific parts of your SAS output are called *locations*. Figure 5 shows the locations that are pertinent to the REPORT procedure output (SAS Institute Inc. 2008a).

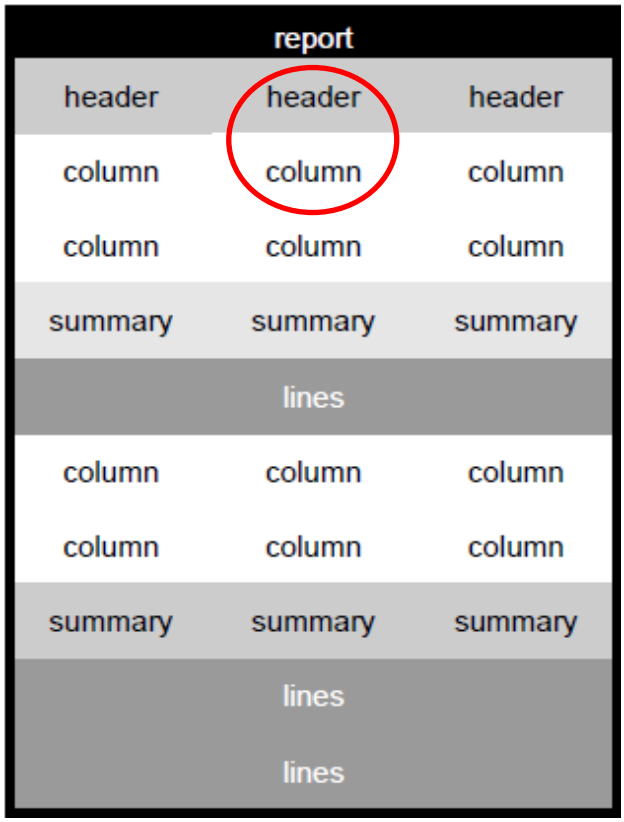


Figure 5. Style Locations for the REPORT Procedure

By default, ODS applies the "header" style element to the "Header" location, and the "data" style element to the "Column" location. Changing the "header" style element, as we have done in the "XLsansPrinter" style, affects all procedure output that uses that style element. Thus, you should carefully consider the impact of changing style elements that are part of a style supplied by SAS.

Style overrides are used to change the appearance of selected portions of your output, and they are supported by the PRINT, TABULATE, and REPORT procedures. They 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 code alters the attributes of the "Header" location:

```
style(Header)=[background=white font_size=10pt just=center]
```

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, and are discussed in the "Centering Selected Data Values" section.

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.

As an example, suppose you want to change the appearance of the column headings in some, but not all, of your output. One solution is to create a new style element based on the original "header" style element, and use a style override to apply the new style element to the "Header" location as needed.

The following code shows you how to create a new style element:

```
proc template;
  define style styles.MysansPrinter;
    parent = styles.sansPrinter;

            ❶          ❷
  style myheader from header /
    background = white
    font_size  = 10pt
    just       = center;
  end;
run; quit;
```

This code does not alter the "header" style element that is supplied by SAS (❷). Instead, it creates a new, user-defined style element named "myheader" (❶) that inherits all of the attributes of the "header" style element. The code then changes the values of the "background", "font_size", and "just" attributes.

The user-defined style element must be specified as a style override before ODS will recognize it. The code below shows how to override the style element used for the "Header" location for only the variables "name" and "age":

```
ods tagsets.ExcelXP style=MysansPrinter ...;
  proc print data=sashelp.class noobs;
    var name age / style(Header)=myheader;
    var sex height weight;
  run; quit;

  proc report data=sashelp.class nowindows;
    column name age sex height weight;
    define name / style(Header)=myheader;
    define age  / style(Header)=myheader;
  run; quit;
ods tagsets.ExcelXP close;
```

The technique of using a CALL DEFINE statement with PROC PREPORT to conditionally apply style overrides is discussed in the "Changing the Background Color of Alternating Rows" and "Traffic Lighting Specific Cells" sections.

Colors Supported by Excel

Excel versions 2002 and 2003 have a limited color palette. Figure 6 shows the default colors. If you plan to view your workbooks using one of these versions of Excel, choose colors that are listed in the default palette. Otherwise Excel maps the unsupported color to a color that is supported.

However, specifying colors is not foolproof. Because each Excel user can customize the color palette, colors are not guaranteed to exist in all instances of Excel.

Note that Excel 2007 and 2010 do not have this restricted color palette.

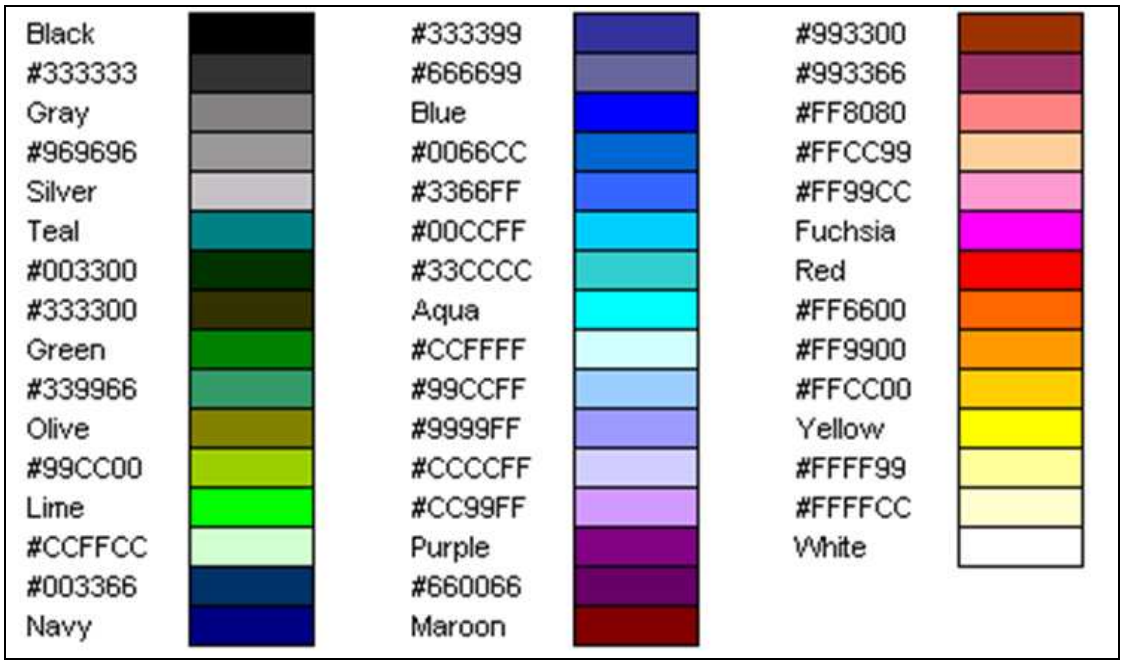


Figure 6. Default Colors Supported by Excel Versions 2002 and 2003

Changing the Background Color of Alternating Rows

Initial execution of the basic SAS code creates the workbook shown in Figure 4. Adding a background color to alternating rows makes the worksheets more attractive and easier to read. This is accomplished by applying a style override using a CALL DEFINE statement. The general syntax of this statement is:

```
call define(column-id | _ROW_, 'attribute-name', attribute-value);
```

Shown below are the modifications to the basic SAS code that are needed to change the background color of odd-numbered rows to blue.

```
proc report data= ... ;
  by RX;
  column PatNo Age SZ Status HX EKG BM SBP DBP;

  define  ...;

  compute PatNo;
    * Placeholder for row highlighting;
    RowNum+1;
    if (mod(RowNum, 2) ne 0)
      then call define(_row_, 'style', 'style=[background=#99ccff]');
  endcomp;

  compute Status;
    * Placeholder for cell highlighting;
  endcomp;

  format RX rx.;

run; quit;
```

First, a new variable named "RowNum" is created in order to keep track of the row number of the data. The MOD function is used to determine whether the value of "RowNum" is evenly divisible by 2. If it is not, a style override changes background color of the row to blue using the automatic variable _ROW_.

Traffic Lighting Specific Cells

The CALL DEFINE statement can be used to apply a style override to specific cells based on a set of rules. This general technique is referred to as "traffic lighting", and is used to emphasize data that meet the rule criteria. In our case, we want to draw attention to patients who died due to cardiovascular disease by changing the cell background color to orange. We use the code below to accomplish this:

```
proc report data= ... ;
  by RX;
  column PatNo Age SZ Status HX EKG BM SBP DBP;

  define  ...;

  compute PatNo;
    * Placeholder for row highlighting;
    RowNum+1;
    if (mod(RowNum, 2) ne 0)
      then call define(_row_, 'style', 'style=[background=#99ccff]');
  endcomp;

  compute Status;
    * Placeholder for cell highlighting;
    if (Status eq 'dead - heart or vascular')
      then call define('Status', 'style', 'style=[background=#ff6600]');
  endcomp;

  format RX rx.;

run; quit;
```

The name of the column is used in the CALL DEFINE statement, instead of the _ROW_ automatic variable.

Centering Selected Data Values

Up to now we have used style overrides to change the appearance of specific cells. But overrides can be applied to an entire column of data using the STYLE option, as discussed in the "Understanding and Using ODS Style Overrides" section. We use the inline format of the STYLE option to center the data values for all columns except "Status" and "EKG".

```
proc report data= ... ;
  by RX;
  column PatNo Age SZ Status HX EKG BM SBP DBP;

  define PatNo  / display style(Column)=[just=center];
  define Age    / display style(Column)=[just=center];
  define SZ     / display style(Column)=[just=center];
  define Status / display;
  define HX     / display format=boolean. style(Column)=[just=center];
  define EKG    / display;
  define BM     / display format=boolean. style(Column)=[just=center];
  define SBP    / display style(Column)=[just=center];
  define DBP    / display style(Column)=[just=center];

  compute PatNo;
    * Placeholder for row highlighting;
    RowNum+1;
    if (mod(RowNum, 2) ne 0)
      then call define(_row_, 'style', 'style=[background=#99ccff]');
  endcomp;

  compute Status;
    * Placeholder for cell highlighting;
    if (Status eq 'dead - heart or vascular')
      then call define('Status', 'style', 'style=[background=#ff6600]');
  endcomp;

  format RX rx.;

run; quit;
```

The result of applying the style overrides is shown in Figure 7.

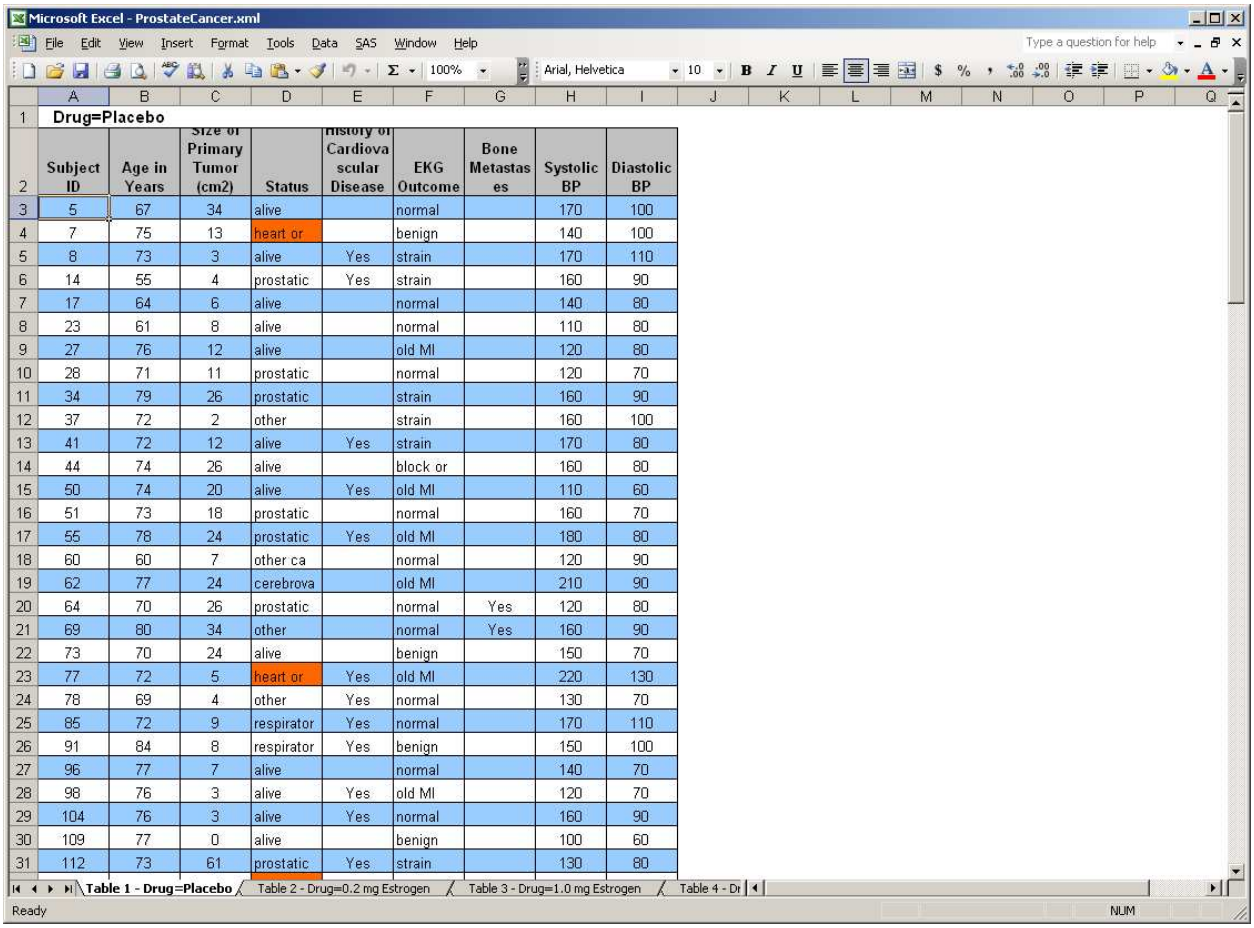


Figure 7. ExcelXP ODS Tagset-Generated Workbook with Style Overrides Applied

WORKING WITH THE EXCELXP TAGSET OPTIONS

As mentioned earlier, 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' ...) ... ;
```

It is important to 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;
```

When specifying *additional* ODS statements as shown above, 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.

Tagset options are supported for **all** SAS procedure output, unlike ODS style overrides, which are supported only by the PRINT, REPORT, and TABULATE procedures.

Using BY Group Values in Worksheet Names

ODS generates a unique name for each worksheet, as required by Excel. Figure 7 shows the worksheet names that result from running the most recent version of the SAS code. There are, however, several tagset options that you can use to alter the names of the worksheets.

The SHEET_INTERVAL option controls the interval at which SAS output is placed into worksheets, and the SHEET_LABEL option is used to specify the prefix to use for the worksheet names. When used together, the current value of the first BY group variable is used in the worksheet name. The following code causes the worksheet names to match those shown in Figure 1:

```
ods tagsets.ExcelXP path='output-directory' file='ProstateCancer.xml'
  style=XLsansPrinter;

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

title;
footnote;

*  One worksheet created for each distinct BY value;

proc report data= ... ;
  *  PROC REPORT code with style overrides applied;
run; quit;

ods tagsets.ExcelXP close;
```

The blank space between the quotation marks for the SHEET_LABEL option suppresses the printing of a prefix. If you want a particular text string to precede the BY variable value, specify that text between the quotation marks.

Suppressing the BY Line Text

BY line text appears in the worksheets because the REPORT procedure is executed with a BY statement. The text is redundant because the BY value is displayed in the worksheet name. To omit the BY line text, specify the SUPPRESS_BYLINES option in the ODS statement that precedes the PROC REPORT code.

```
ods tagsets.ExcelXP path='output-directory' file='ProstateCancer.xml'
  style=XLsansPrinter;

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

title;
footnote;

*  One worksheet created for each distinct BY value;

proc report data= ... ;
  *  PROC REPORT code with style overrides applied;
run; quit;

ods tagsets.ExcelXP close;
```

Do not attempt to use the NOBYLINE *system* option, as this disables BY group processing in the ExcelXP tagset even though the SHEET_INTERVAL tagset option is set to "bygroup".

Adding Excel AutoFilters

An Excel AutoFilter enables you to filter, or subset, the data that is being displayed in a worksheet. A column of data containing an AutoFilter is indicated by an arrow button in the header cell of that column. For example, columns B through G in Figure 1 contain an AutoFilter.

Suppose you want to view only records for patients that do not have a history of cardiovascular disease. You click the AutoFilter button on the "History of Cardiovascular Disease" column, and then select **(Blanks)**, as illustrated in Figure 8. All records in the worksheet that do not have blank as a value in the "History of Cardiovascular Disease" column are hidden. The data is still present in the worksheet, but it is not displayed due to the filtering.

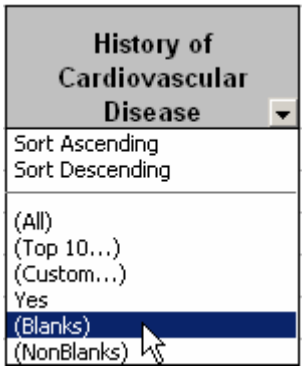


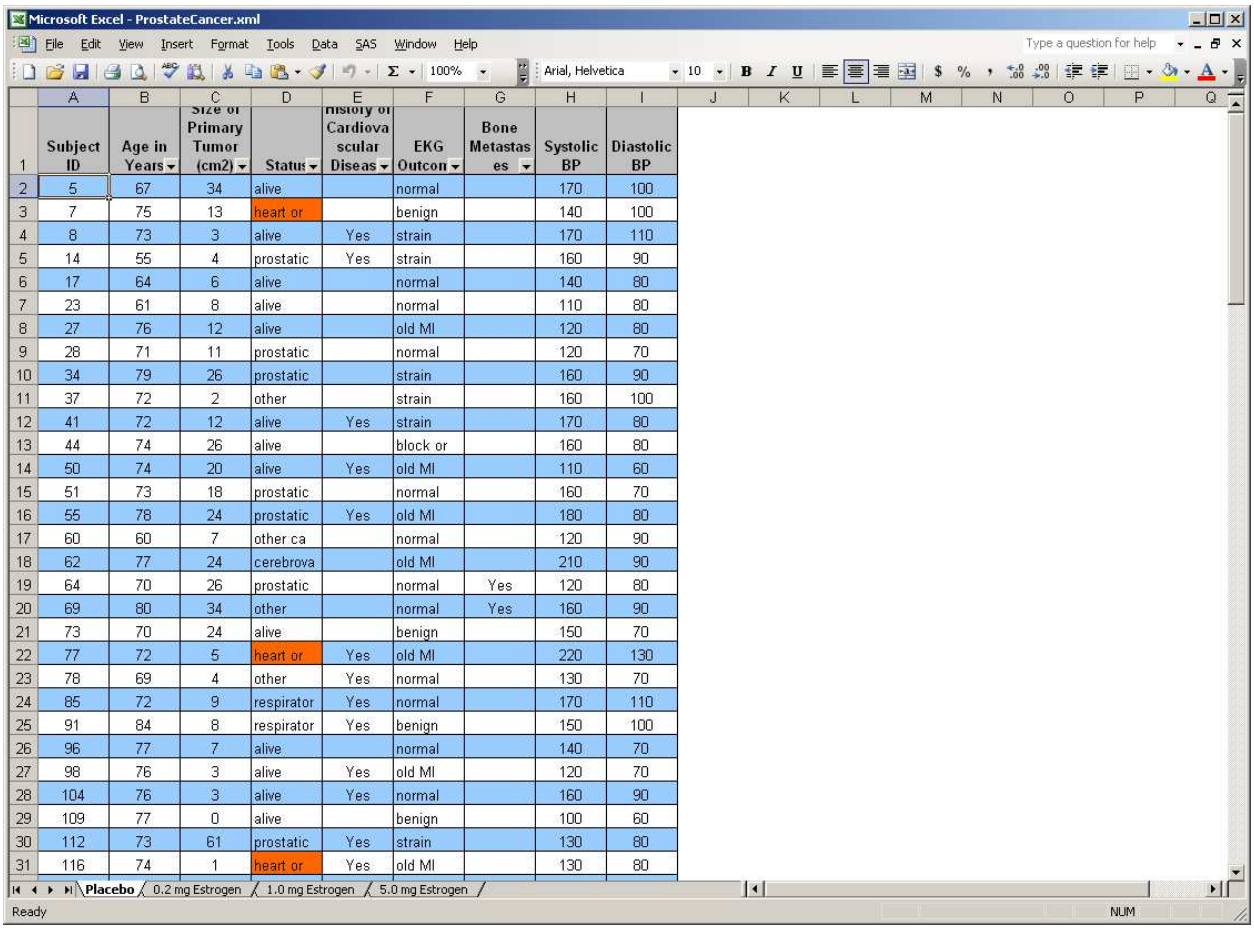
Figure 8. Excel AutoFilter

AutoFilter selections are additive. If you want to see patients without a history of cardiovascular disease and who died as a result of a heart or vascular cause, click the AutoFilter button on the "Status" column and then select **dead – heart or vascular**. To clear an AutoFilter select **(All)** from the drop-down list.

You can control which columns have AutoFilters by specifying "all", "none", or a range of contiguous columns in the AUTOFILTER tagset option. We want AutoFilters on columns B through G, so we specify the value "2–7".

```
ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label=' '
                            suppress_bylines='yes'
                            autofilter='2-7');
```

Figure 9 shows the workbook created by running the code with style overrides and tagset options applied.



	A	B	C	D	E	F	G	H	I
1	Subject ID	Age in Years	Size of Primary Tumor (cm2)	Status	History of Cardiovascular Diseas	EKG Outcon	Bone Metastases	Systolic BP	Diastolic BP
2	5	67	34	alive		normal		170	100
3	7	75	13	heart or		benign		140	100
4	8	73	3	alive	Yes	strain		170	110
5	14	55	4	prostatic	Yes	strain		160	90
6	17	64	6	alive		normal		140	80
7	23	61	8	alive		normal		110	80
8	27	76	12	alive		old MI		120	80
9	28	71	11	prostatic		normal		120	70
10	34	79	26	prostatic		strain		160	90
11	37	72	2	other		strain		160	100
12	41	72	12	alive	Yes	strain		170	80
13	44	74	26	alive		block or		160	80
14	50	74	20	alive	Yes	old MI		110	60
15	51	73	18	prostatic		normal		160	70
16	55	78	24	prostatic	Yes	old MI		180	80
17	60	60	7	other ca		normal		120	90
18	62	77	24	cerebrova		old MI		210	90
19	64	70	26	prostatic		normal	Yes	120	80
20	69	80	34	other		normal	Yes	160	90
21	73	70	24	alive		benign		150	70
22	77	72	5	heart or	Yes	old MI		220	130
23	78	69	4	other	Yes	normal		130	70
24	85	72	9	respirator	Yes	normal		170	110
25	91	84	8	respirator	Yes	benign		150	100
26	96	77	7	alive		normal		140	70
27	98	76	3	alive	Yes	old MI		120	70
28	104	76	3	alive	Yes	normal		160	90
29	109	77	0	alive		benign		100	60
30	112	73	61	prostatic	Yes	strain		130	80
31	116	74	1	heart or	Yes	old MI		130	80

Figure 9. ExcelXP ODS Tagset-Generated Workbook with Style Overrides and Tagset Options Applied

Adjusting Column Widths

Some of the column widths are too narrow, causing data to be obscured, or causing unattractive wrapping of column headings (Figure 9). The ExcelXP tagset uses the following formula to compute the approximate column width, sometimes resulting in less than perfect widths:

```
ColumnWidth = WIDTH_POINTS * ABSOLUTE_COLUMN_WIDTH * WIDTH_FUDGE
```

You can adjust column widths using any combination of these three tagset options. If you do not specify values for these options, they are determined for you, as explained in Table 2.

Tagset Option	What Determines the Option Value
WIDTH_POINTS	Font metrics.
ABSOLUTE_COLUMN_WIDTH	Numeric columns: the largest number of characters in any cell of the column, including the column heading. Character columns: the length of the character variable, or the largest number of characters in a cell in the column, whichever is larger.
WIDTH_FUDGE	A multiplier that can be used to make small adjustments to *all* column widths. Default value is 0.75. Increasing the value widens the columns, while decreasing the value narrows the columns.

Table 2. Tagset Options That Control Column Widths

For our data, specifying values for each column using ABSOLUTE_COLUMN_WIDTH improves the display. To determine values for ABSOLUTE_COLUMN_WIDTH that you can specify, first resize the columns in Excel until they are an appropriate width, or use the automatic resize columns feature.

To automatically resize all columns to fit their contents, click the Excel "Select All" button to select all columns (Figure 10). Next, click (Excel 2007 and 2010) **`Home ➨ Format ➨ AutoFit Column Width`**, or (Excel 2002 and 2003) **`Format ➨ Column ➨ AutoFit Selection.`** If Excel does not resize all columns appropriately, select the AutoFit sequence again while all columns remain selected.

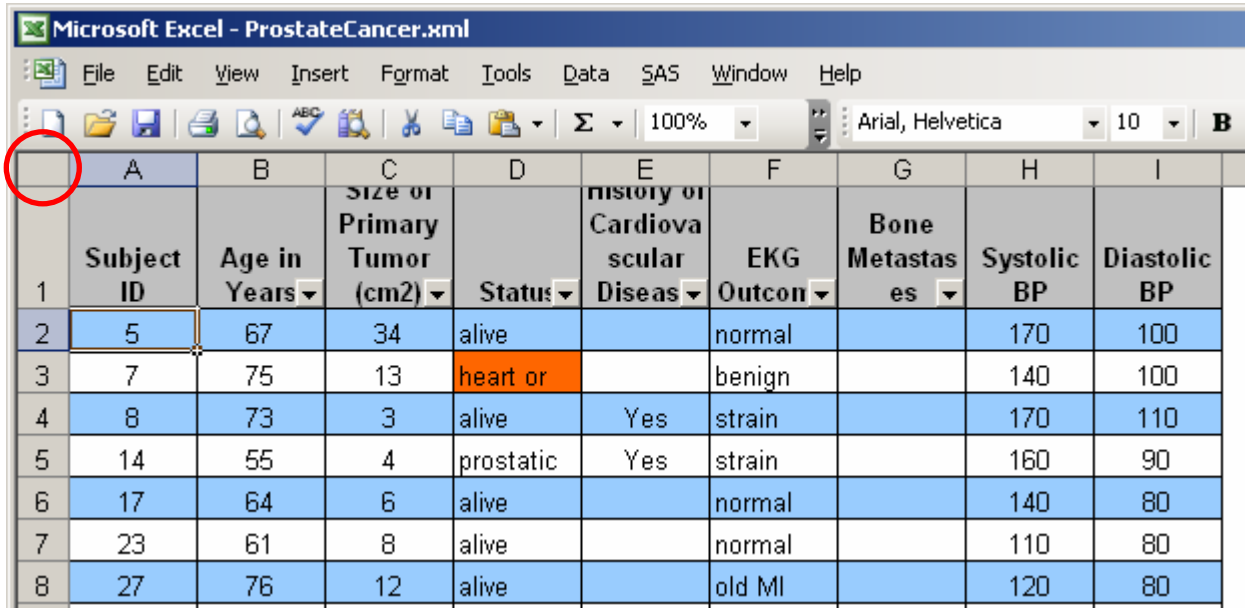


Figure 10. Location of the Excel "Select All" Button for all Versions of Excel

To display the values to use for ABSOLUTE_COLUMN_WIDTH click (Excel 2007 and 2010) **`Home ➨ Format ➨ Column Width`**, or (Excel 2002 and 2003) **`Format ➨ Column ➨ Width.`** Record the value for each column, and specify them as a comma-separated list of values for the ABSOLUTE_COLUMN_WIDTH option.

```
ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label=' '
                            suppress_bylines='yes'
                            autofilter='2-7'
                            absolute_column_width='7.14,10.43,18.29,23.57,18.2,
                                                   27.71,14.43,7.29,8');
```

If the tagset computes widths that are still not satisfactory, you can make small adjustments to *all* columns widths by using the WIDTH_FUDGE option. For our data, a value of 0.58 produces more appropriate column widths than the default value of 0.75.

```
ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label=' '
                            suppress_bylines='yes'
                            autofilter='2-7'
                           absolute_column_width='7.14,10.43,18.29,23.57,18.2,
                                                  27.71,14.43,7.29,8'
                           width_fudge='0.58');
```

For alternate techniques of determining appropriate values for the ABSOLUTE_COLUMN_WIDTH option, refer to this author's earlier papers (DelGobbo 2006, 2009, 2010).

Frozen Column Headings

The REPORT procedure output shown in Figure 1 contains over 200 rows, and scrolling down past row 31 causes the column headings to move off the viewable screen. You can correct this problem by using the FROZEN_HEADERS tagset option to "freeze" the rows that you want to use as column headings.

Valid values for this option are either "yes" or an integer. If you specify "yes", the tagset attempts to automatically determine the rows to freeze. If you do not like the results, you can explicitly specify the rows to freeze using an integer. For example, to freeze rows 1 through 3, specify `frozen_headers='3'`. In our case using "yes" results in good output, and as a result, the code is:

```
ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label=' '
                            suppress_bylines='yes'
                            autofilter='2-7'
                           absolute_column_width='7.14,10.43,18.29,23.57,18.2,
                                                  27.71,14.43,7.29,8'
                           width_fudge='0.58'
                           frozen_headers='yes');
```

THE FINAL SAS CODE

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

```
*;
*  Create formats to make drug codes
*  and Boolean values more user-friendly
*;

proc format;

  value rx 1 = 'Placebo'
           2 = '0.2 mg Estrogen'
           3 = '1.0 mg Estrogen'
           4 = '5.0 mg Estrogen';

  value boolean 0 = ' '
                1 = 'Yes';
run; quit;

ods listing close;

ods tagsets.ExcelXP path='output-directory' file='ProstateCancer.xml'
  style=XLsansPrinter;

ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label=' '
                            suppress_bylines='yes'
                            autofilter='2-7'
                            absolute_column_width='7.14,10.43,18.29,23.57,18.29,
                                                   27.71,14.43,7.29,8'
                            width_fudge='0.58');
```

```
title;
footnote;

*  One worksheet created for each distinct BY value;

proc report data=sample.ProstateCancer nowindows split='*';
  by RX;
  column PatNo Age SZ Status HX EKG BM SBP DBP;

  define PatNo  / display style(Column)=[just=center];
  define Age    / display style(Column)=[just=center];
  define SZ     / display style(Column)=[just=center];
  define Status / display;
  define HX     / display format=boolean. style(Column)=[just=center];
  define EKG    / display;
  define BM     / display format=boolean. style(Column)=[just=center];
  define SBP    / display style(Column)=[just=center];
  define DBP    / display style(Column)=[just=center];

  compute PatNo;
    * Placeholder for row highlighting;
    RowNum+1;
    if (mod(RowNum, 2) ne 0)
      then call define(_row_, 'style', 'style=[background=#99ccff]');
  endcomp;

  compute Status;
    * Placeholder for cell highlighting;
    if (Status eq 'dead - heart or vascular')
      then call define('Style', 'style', 'style=[background=#ff6600]');
  endcomp;

  format RX rx.;

run; quit;

ods tagsets.ExcelXP close;
```

SAS SERVER TECHNOLOGY

You can incorporate dynamically-generated SAS output into Excel using the Application Dispatcher or SAS® Stored Process Server. 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 software 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 11 shows a Web page that can deliver SAS output directly to Excel, using a Web browser as the client.

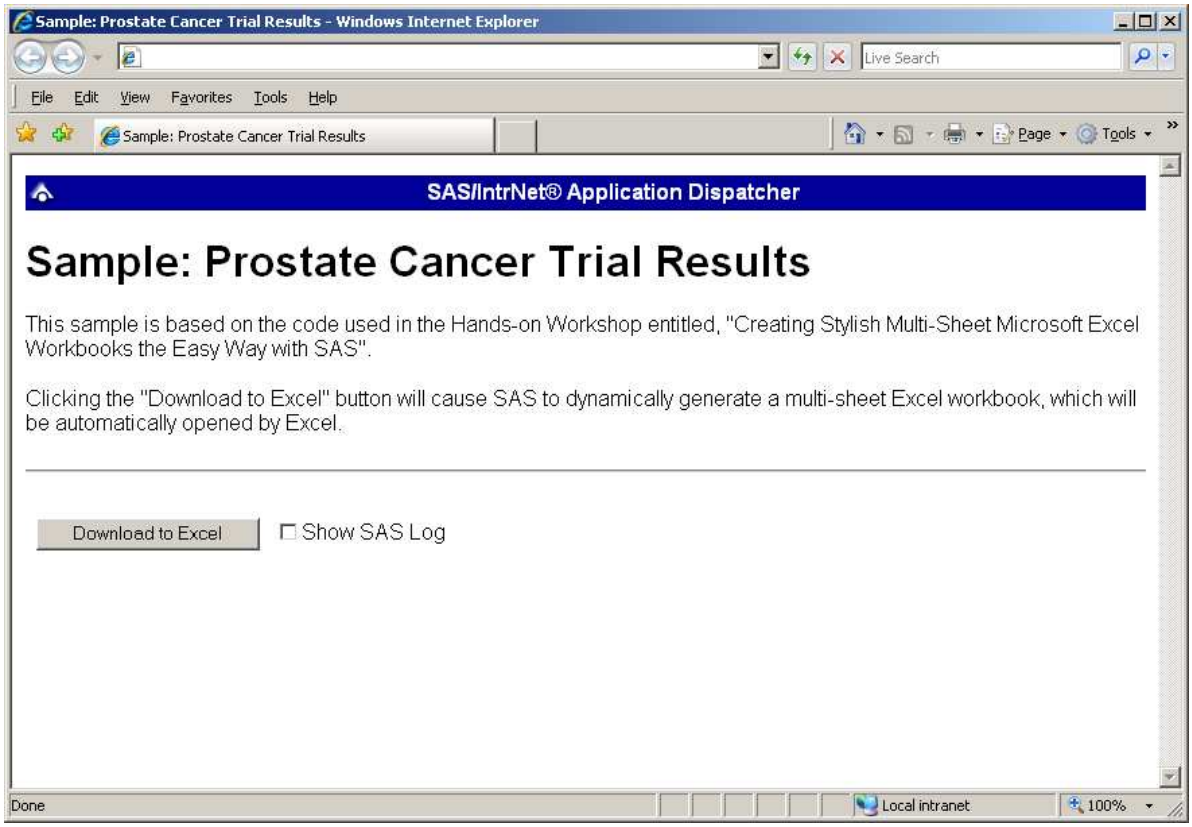


Figure 11. Web Page to Drive a SAS/IntrNet Application

Clicking **Download to Excel** executes a slightly modified version of the SAS code that we have been working on. The modifications are as follows:

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

ods listing close;
ods tagsets.ExcelXP file=_webout style=XLsansPrinter;
  *  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 populates the file name field in the "File Download" dialog box. As you can see in Figure 12, the file name appears as "ProstateCancer.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.

The reserved fileref _WEBOUT is automatically assigned by the SAS

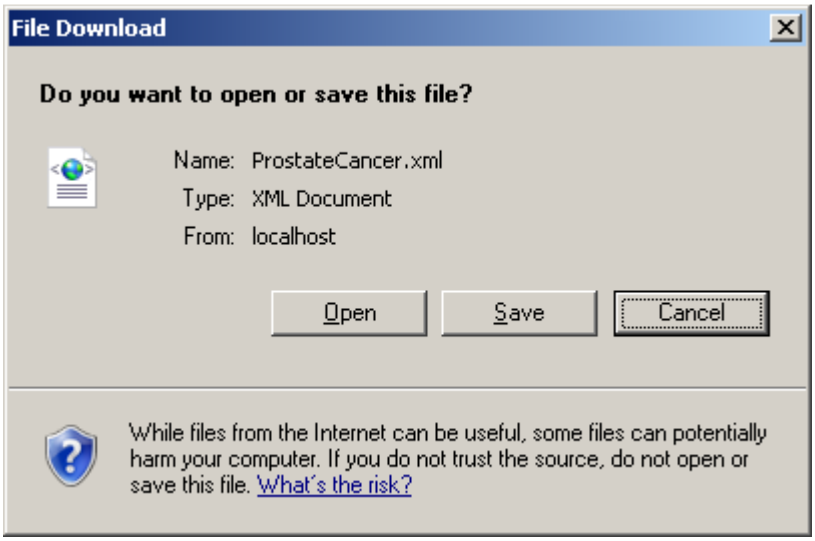


Figure 12. File Download Dialog Box

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 the **Download to Excel** button on the Web page and are presented with the "File Download" dialog box (Figure 12), you can click **Open** to immediately open your SAS output using Excel, or **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. 2008b, 2009c), as well as this author's earlier papers (DelGobbo 2002, 2003, 2004).

CONCLUSION

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

CODE FOR CREATING THE XLSANSPRINTER STYLE

```
proc template;
  define style styles.XLsansPrinter;
    parent = styles.sansPrinter;

    /*  Change attributes of the column headings  */

    style header from header /
      font_size = 10pt
      just      = center
      vjust     = bottom;
  end;
run; quit;
```

DIAGNOSING EXCEL LOAD ERRORS

When you open a malformed XML file using Excel, you are presented with an error dialog box containing a message like this:

```
This file cannot be opened because of errors. Errors are listed in: C:\Documents and
Settings\userid\Local Settings\Temporary Internet Files\Content.MSO\404A54C0.log.
```

On most systems, the log file created by Excel is stored in a hidden directory, so you cannot navigate to it using Microsoft Windows Explorer. You can, however, navigate to the directory from a command prompt (cmd.exe) by typing this text and pressing the ENTER key (ignore the line wrapping):

```
%SYSTEMROOT%\explorer.exe /e, "%USERPROFILE%\Local Settings\Temporary Internet
Files\Content.MSO"
```

The malformed XML is often created because of a typographical error in the name of the style element used in a style override, or because a style element was specified that is not part of the style definition. Here is an example of the first case, where the "data_bold" style element is defined in the "MyStyle" style, but it is misspelled in the style override as "date_bold":

```
ods tagsets.ExcelXP file='MyWorkbook.xml' style=MyStyle ...;

...

define MyVar / display order style(Column)=date_bold;

...

ods tagsets.ExcelXP close;
```

Here is an example of the second case, were the "data_bold" style element is spelled correctly, but it is not defined in the "sansPrinter" style that is supplied by SAS. The style element exists in the user-defined "XLsansPrinter" style, but you forgot to change the style name from "sansPrinter" to "XLsansPrinter":

```
ods tagsets.ExcelXP file='MyWorkbook.xml' style=sansPrinter ...;

...

define MyVar / display order style(Column)=data_bold;

...

ods tagsets.ExcelXP close;
```

In both instances, the Excel log file contains the following information, indicating that "unknown" is an invalid value for the "StyleID" attribute of the "Cell" tag:

```
XML ERROR in Table
REASON: Bad Value
FILE: C:\HOW\DelGobbo\MyWorkbook.xml
GROUP:  Row
TAG:  Cell
ATTRIB: StyleID
VALUE:  unknown
```

Open the file "MyWorkbook.xml" using a text editor and you will see instances of the incorrect value being used:

```
<Cell ss:StyleID="unknown" ... >
```

REFERENCES

Andrews, D.F. and Herzberg, A.M. 1985: *Data: A Collection of Problems from Many Fields for the Student and Research Worker*. 261-274. New York: Springer-Verlag.

Bailar III, John C. and Byar, David P. 1970: "Estrogen Treatment for Cancer of the Prostate: Early Results with 3 Doses of Diethylstilbestrol and Placebo". *Cancer*. 26(2):257-261.

Byar, D.P. and Green, S.B. 1980: "The Choice of Treatment for Cancer Patients Based on Covariate Information: Application to Prostate Cancer". *Bull Cancer*. 67(4):477-490.

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. 2006. "Creating *AND* Importing Multi-Sheet Excel Workbooks the Easy Way with SAS®". *Proceedings of the Thirty-First Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi31/115-31.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.

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

Microsoft Corporation. 2011. "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. *SAS/IntrNet® 9.2: Application Dispatcher*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/dispatch/59547/HTML/default/main_contents.htm.

SAS Institute Inc. 2009a. "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. 2009b. *SAS® 9.2 Output Delivery System: User's Guide*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/a002565239.htm.

SAS Institute Inc. 2009c. *SAS® 9.2 Stored Processes: Developer's Guide*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/stpug/61271/HTML/default/a003152554.htm.

SAS Institute Inc. 2009d. "Uses of the Results Window". *SAS® 9.2 Language Reference: Concepts*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/lrcon/61722/HTML/default/a003107493.htm#a003107660.

SAS Institute Inc. 2011. "ODS MARKUP Resources". Available at http://support.sas.com/rnd/base/topics/odsmarkup/.

ACKNOWLEDGMENTS

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

RECOMMENDED READING

DelGobbo, Vincent. 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.

DelGobbo, Vincent. 2010. "Vince DelGobbo's ExcelXP Tagset Paper Index". Available at http://www.sas.com/events/cm/867226/ExcelXPPaperIndex.pdf.

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

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.