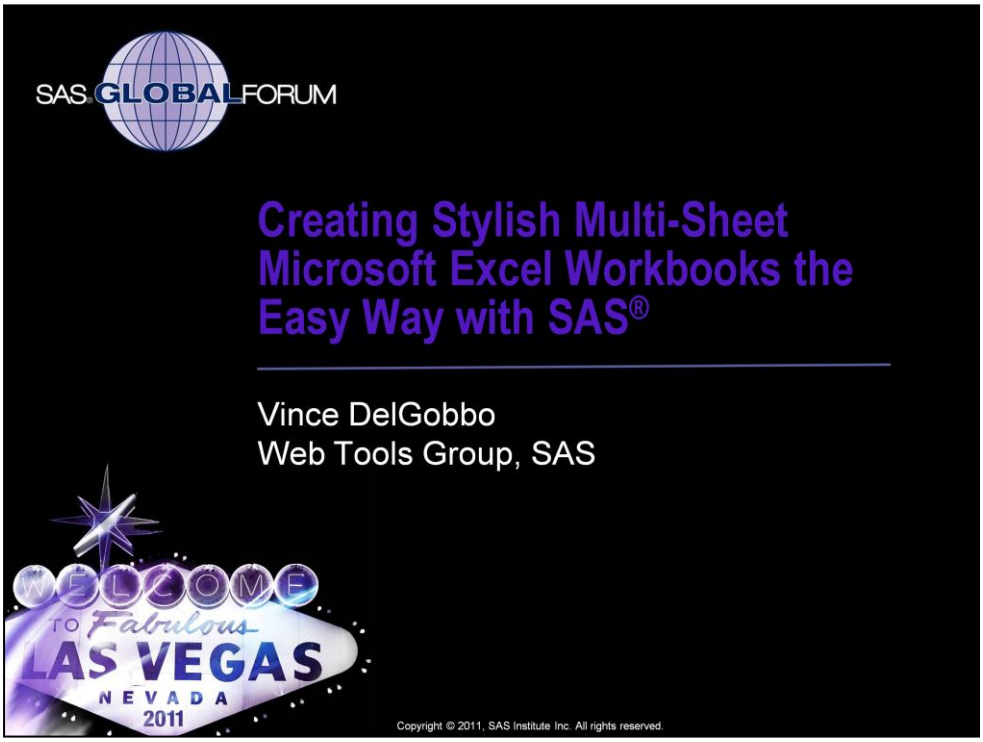


SAS Global Forum 2011

April 4 – 7, 2011

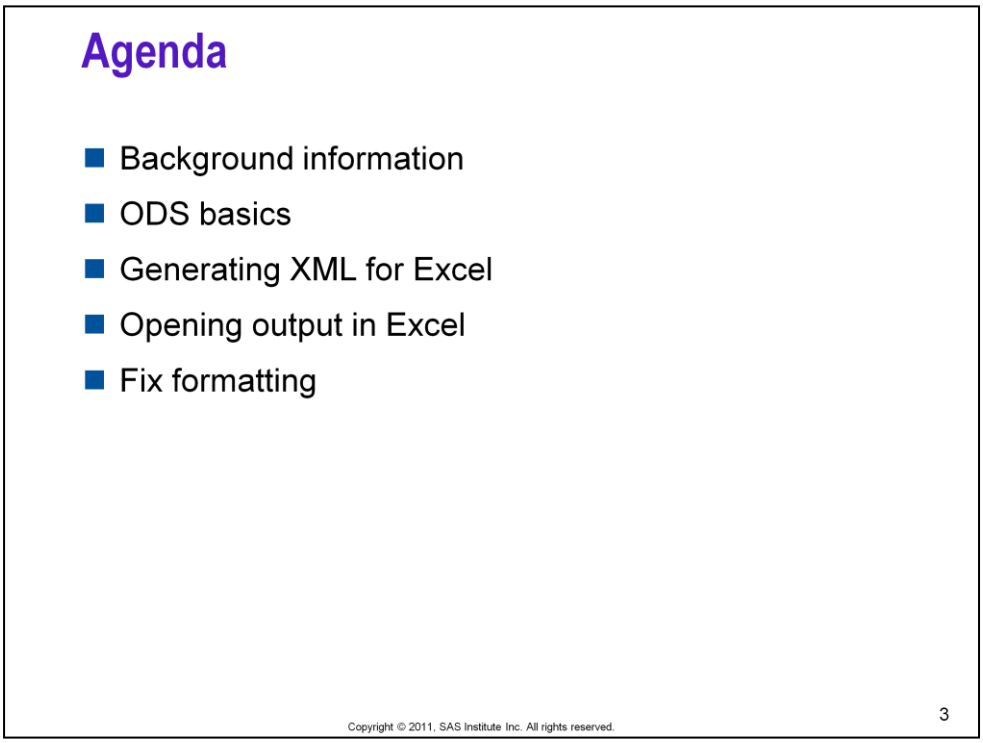
Caesar's Palace

Las Vegas, NV

A special "thank you" to Zul Habib and Ann Stephan for inviting me to present this topic, and to Chris Barrett of SAS Institute Inc. for his valuable contributions to the accompanying paper.

Goals

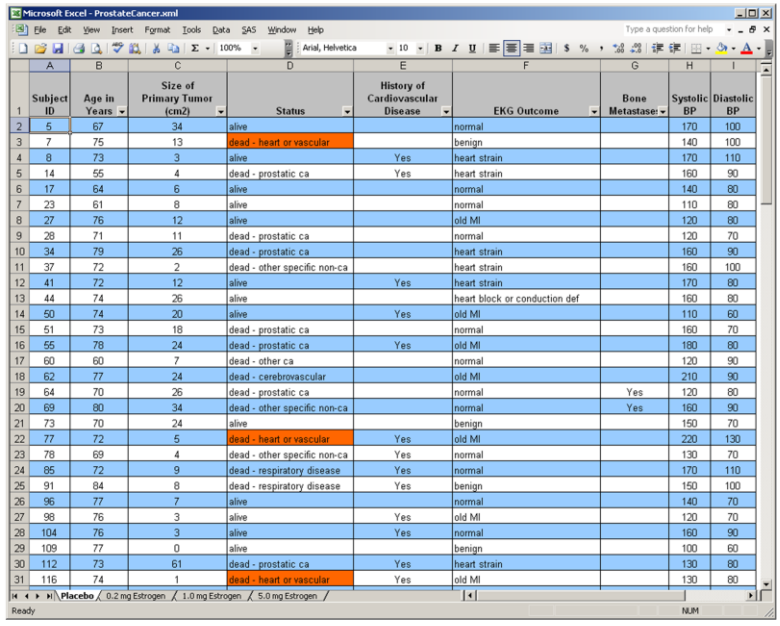
- Give you something you can use TODAY
- Integrate SAS output w/ Excel



Software Requirements

- Base SAS, ***any*** operating system.
- SAS 9.1.3 or later.
- **Updated version** of the ExcelXP tagset (see the accompanying paper for details).
- Microsoft Excel XP (a.k.a. Microsoft Excel 2002) or later.

PROC REPORT Output → Excel



	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

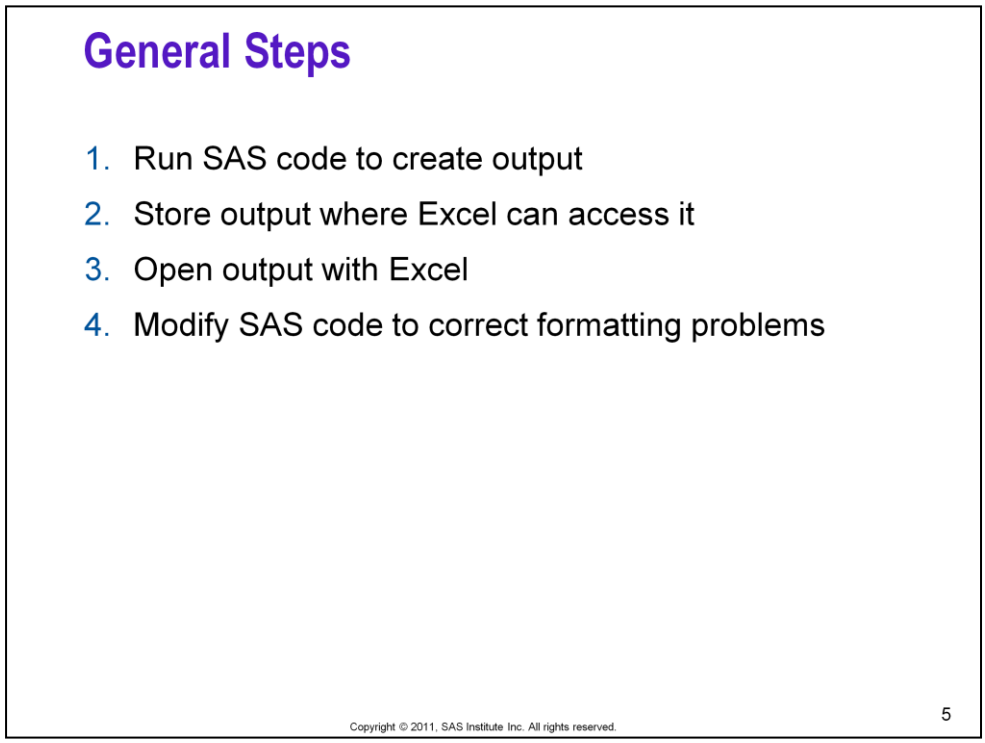
4

The workbook was created using PROC REPORT and contains 4 worksheets containing randomized clinical trial data comparing four treatments for patients with prostate cancer.

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.

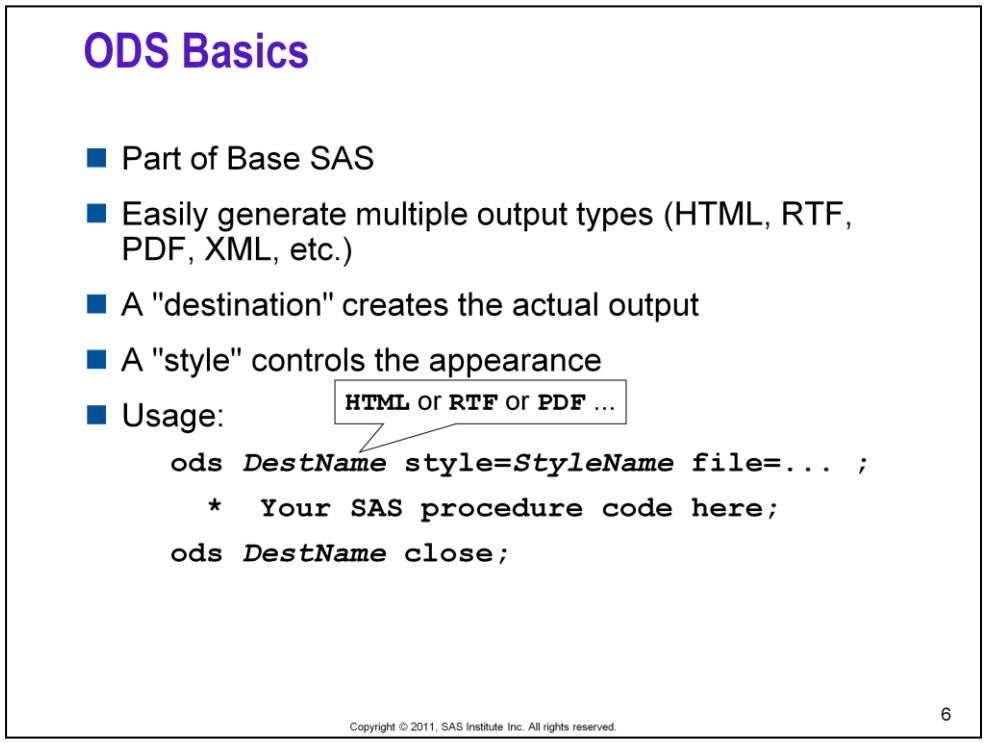
This workbook was generated completely by ODS – there was no "hand-editing" of the Excel workbook. Furthermore, SAS can generate this type of output regardless of the platform – Windows, UNIX, OpenVMS, or z/OS.

The techniques used to highlight rows and traffic light cells also work with the HTML, RTF, and PDF destinations.



We use ODS to create an XML file that is stored in a location where Excel can access it. In your production system, SAS and Excel may reside on two different machines. Thus, you may have to make use of network drives, FTP or some other means to move the SAS output to a location so that Excel can access it.

Then the ODS output is opened using Excel. If you have ever done this before, you have probably encountered formatting problems. We fix those problems, and then explore techniques to instruct ODS to create output that Excel is happy with.



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 (colors, fonts, border lines, etc.).

Both a destination and a style are needed to generate output. If you do not specify a style, the style named "Default", which is shipped with Base SAS software, will be used.

We use a type of ODS destination called a "tagset". ODS tagsets can be modified to meet your specific needs using the TEMPLATE procedure. And you can use the TEMPLATE procedure to create your own tagsets.

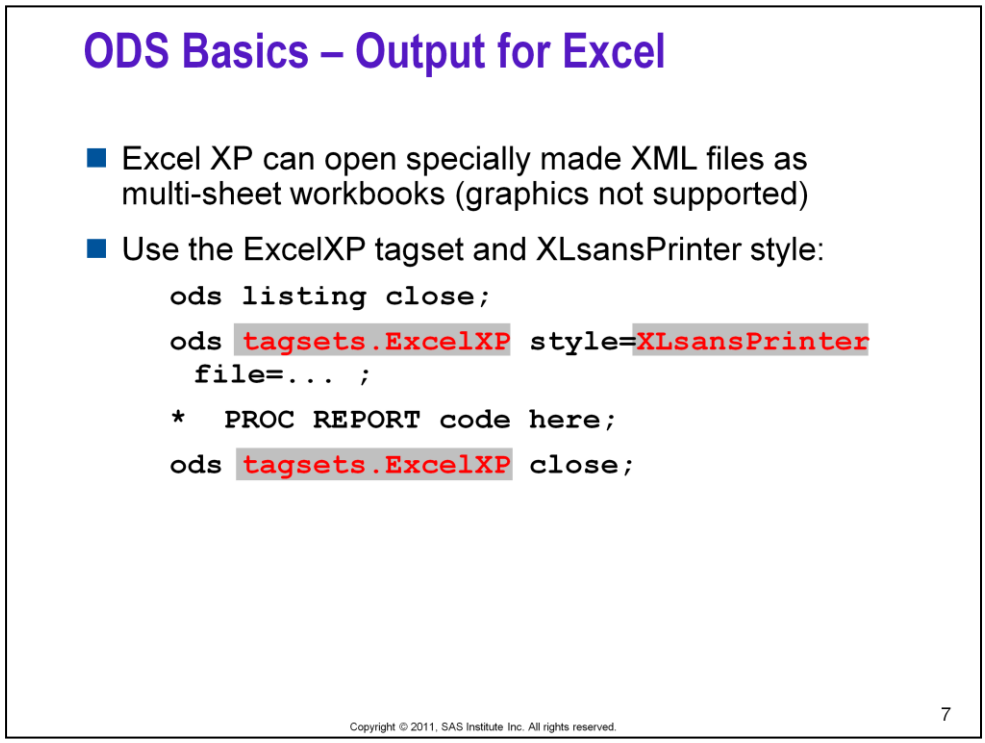
References:

SAS 9.1:

http://support.sas.com/onlinedoc/913/docMainpage.jsp?_topic=odsug.hlp/a001020001.htm

SAS 9.2:

http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/a001020001.htm

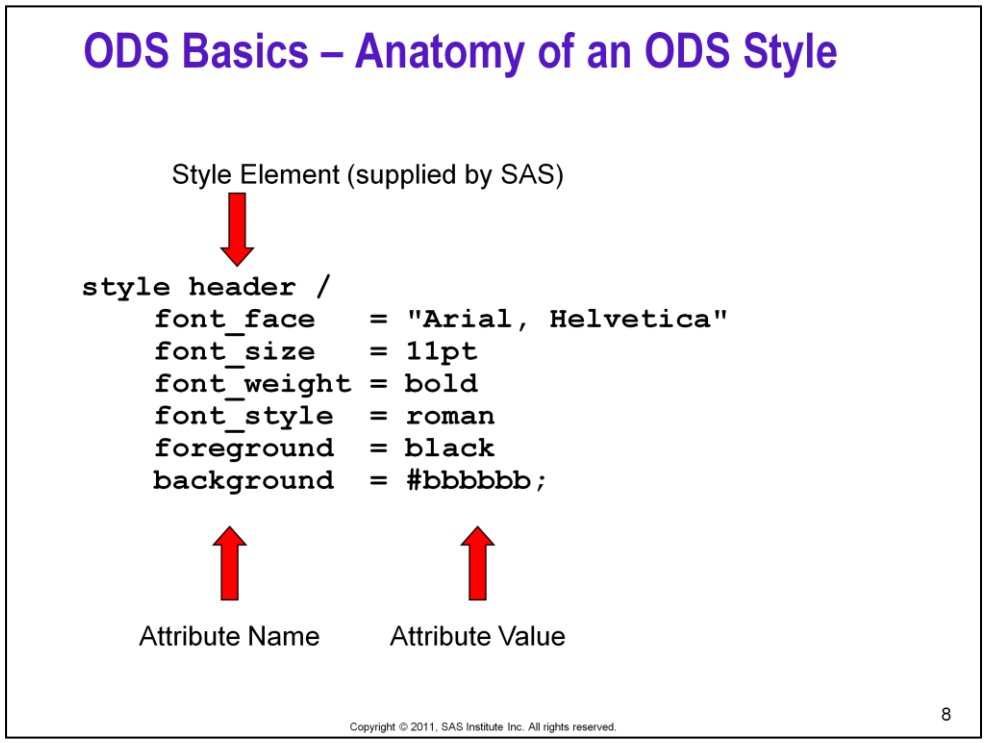


We use the ODS tagset named ExcelXP to create XML output that can be opened using Microsoft Excel XP and later. When opened with Excel, the XML file will be rendered as a multi-sheet Excel workbook. Additionally, we will use a user-defined ODS style named XLsansPrinter to provide a look similar to the sansPrinter style that is shipped with Base SAS software.

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.

Reference:

http://msdn2.microsoft.com/en-us/library/aa140066(office.10).aspx



ODS styles control all aspects of the appearance of the output, and Base SAS software ships with over 40 different styles. A style is composed of *style elements,* each of which controls a particular part of the output. For example, styles supplied by SAS contain a style element named "header" that controls the appearance of column headings.

Style elements consist of collections of *style attributes*, such as the background color and font size. The definition of the "header" style element that is supplied by SAS might look like what is show here.

You can use the TEMPLATE procedure, which is part of Base SAS, to modify an existing style or to create a new style.

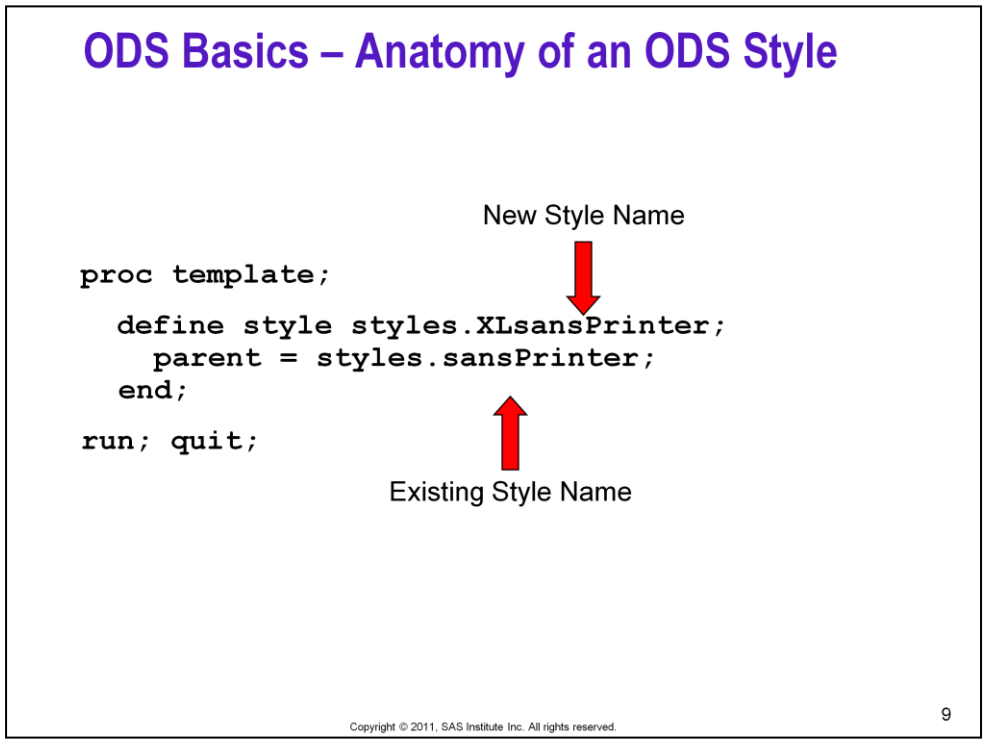
References:

SAS 9.1:

http://support.sas.com/onlinedoc/913/docMainpage.jsp?_topic=odsug.hlp/a002565239.htm

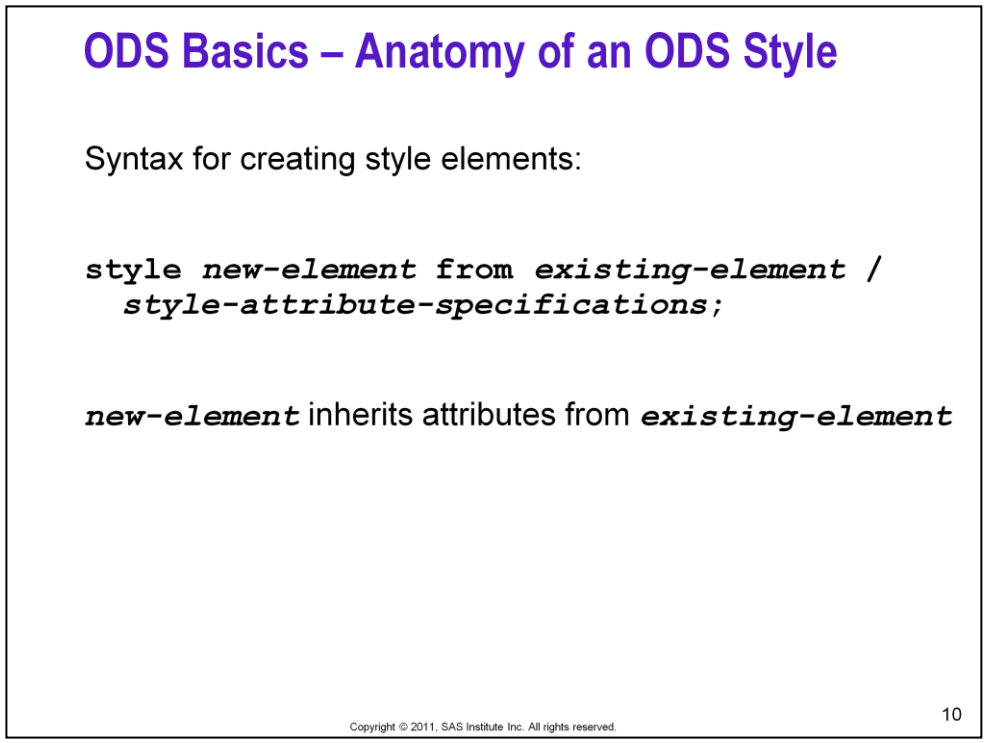
SAS 9.2:

http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/a002565239.htm

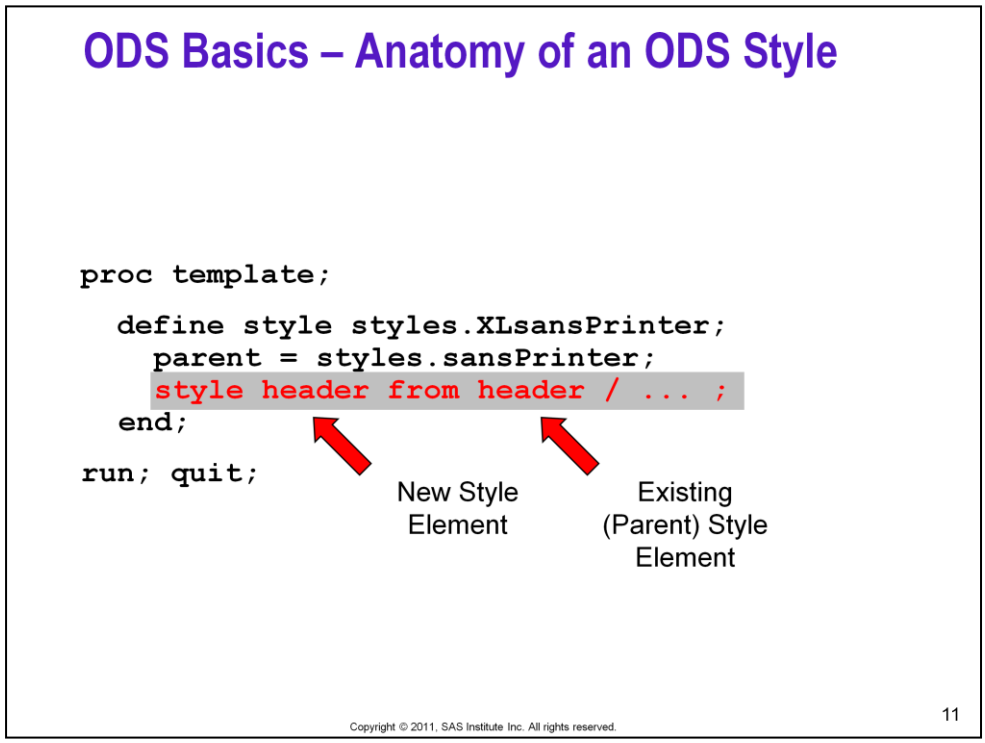


Although you can use the TEMPLATE procedure to create a new style without copying an existing style, it is usually easier to copy an existing style that is close to what you want, and then make modifications to the copy. This code creates the user-defined XLsansPrinter style by copying the ODS style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

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.

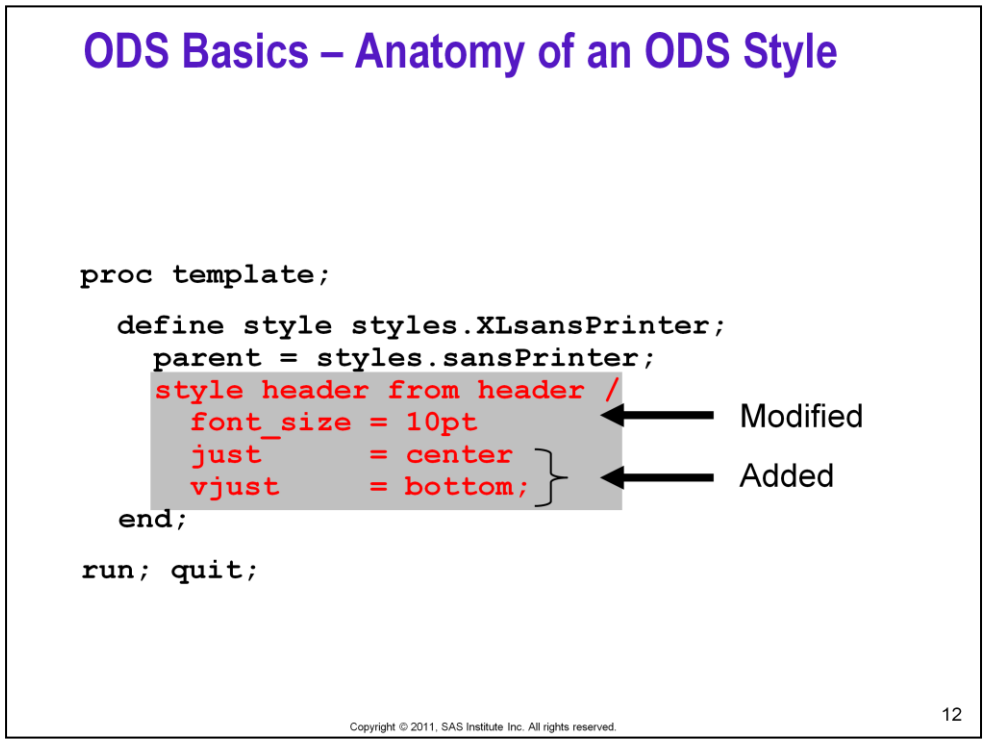


Style elements are created using the "style" statement, which follows this general format.

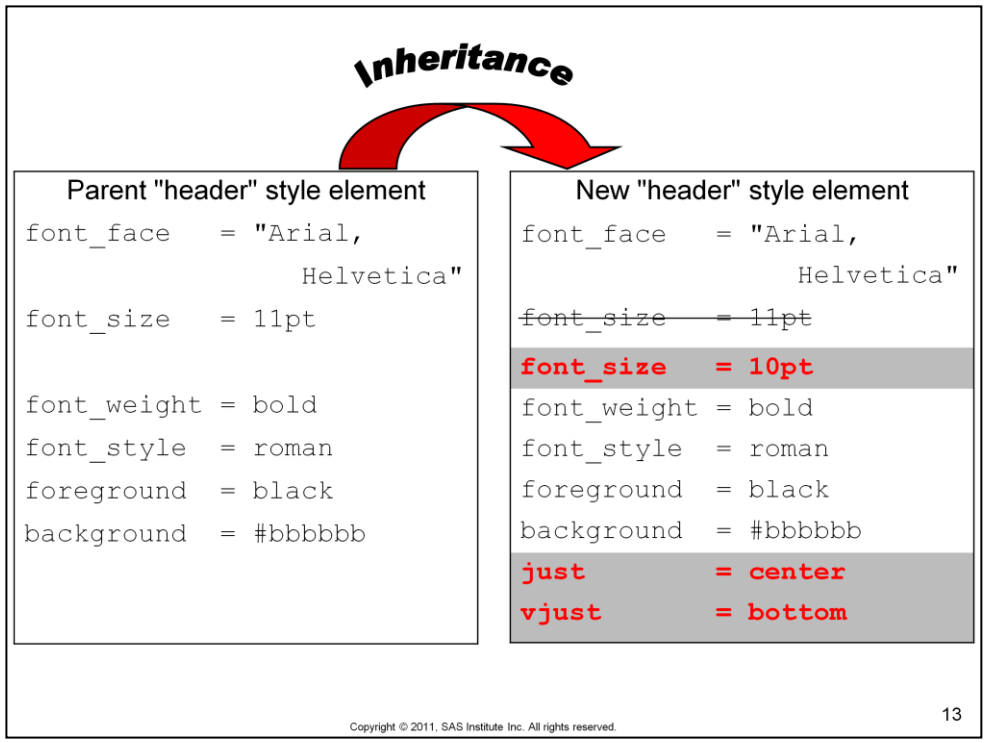


Here is an example of how you would modify the "header" style element that is supplied by SAS. The new "header" style element inherits all the style attributes of the existing (parent) "header" style element shown earlier.

This style element is used to control the appearance of column headings in procedure output. Thus, any time the XLsansPrinter style is used, the appearance of the column headings will be controlled by the attributes specified in this style element.

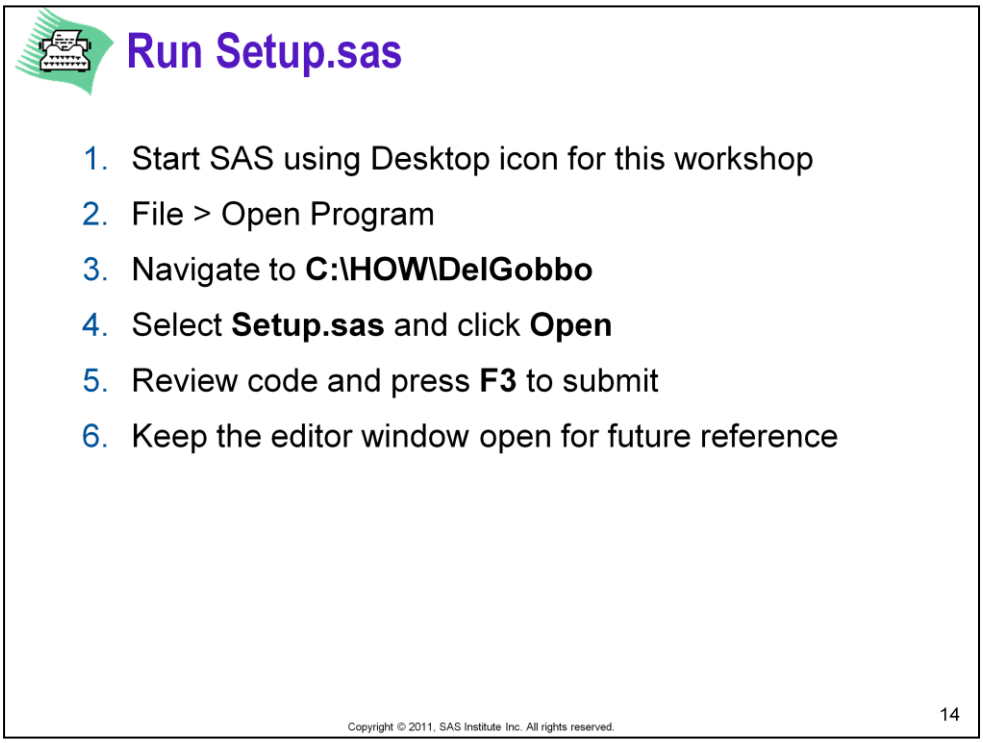


This code illustrates how you can change the value of an existing style attribute ("font_size"), and also add 2 new attributes ("just" and "vjust"). All other attributes of this style element, such as the foreground color, are inherited from the parent "header" style element, and remain unchanged.



This is a graphical representation of the code listed on the previous slide.

The new "header" style element inherits all the attributes of the parent "header" element. 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 attributes inherited from the existing "header" style element, and remain unchanged.



The SAMPDIR global macro variable specifies the directory containing our sample code and data, as well as the ODS-generated XML output.

The program assigns a SAS library (SAMPLE) for the input data and one (MYLIB) for the output ODS tagsets and styles that we create. While it is OK to use the WORK or SASUSER libraries to temporarily store your tagsets and styles, it is a good idea use a different, permanent library that is publicly accessible if you want to make them available to others, or want to reuse them in future programs.

The ODS PATH statement 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 creates and store tagsets and styles in the directory that is associated with the MYLIB library.

The ODS ExcelXP tagset has undergone many revisions since SAS 9.1 was released, so we will import a newer copy and store it in the MYLIB library. Be sure to use a recent version of the ExcelXP tagset in your production code. (See the accompanying paper for details.)

The user-defined XLsansPrinter style, which produces output similar in appearance to the sansPrinter style that is supplied by SAS, is created. Note that the "header" style element supplied by SAS is modified.

Finally, the SAS formats "rx and "boolean" are created.

Sample SAS Data – Prostate Cancer

Variable Name	Typical Values
RX	1, 2, 3, or 4
PatNo	1 – 506
Age	48 – 89
SZ	0 – 69
Status	alive, dead – prostatic ca
BM	0 or 1
HX	0 or 1
EKG	normal, heart strain
SPB	80 – 300
DBP	40 – 180

15

The "ProstateCancer" SAS table contains clinical trial data that is displayed using PROC REPORT.

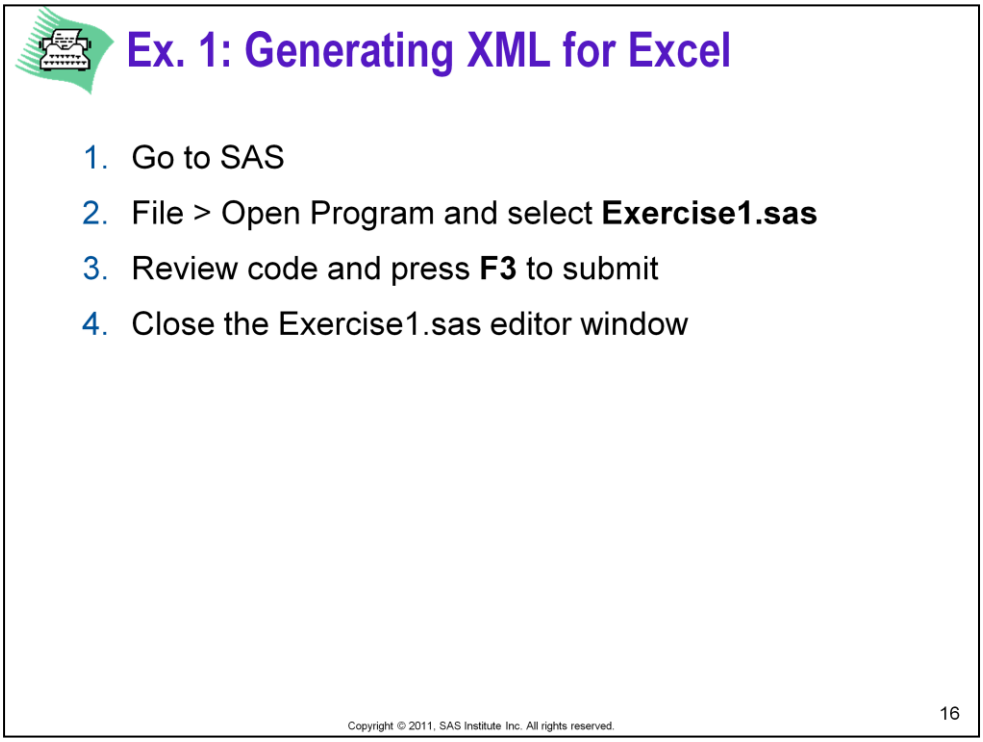
The numeric variable "RX" represents the drug that the patient received, and a SAS format is applied to display the actual treatment. This variable is used in a BY statement in our PROC REPORT code, and the formatted BY group values are used in the resulting worksheet names.

"PatNo" and "Age" are the patient's identification number and age, respectively, and "SZ" represents the size of the primary tumor in cm^2.

"Status" indicates whether the patient is alive, or dead, and if dead, the cause of death.

"BM" and "HX" are Boolean variables that indicate whether or not the patient had bone metastases or a history of cardiovascular disease, respectively.

"EKG" represents the result of an EKG test, and "SPB" and "DBP" are the systolic and diastolic blood pressure, respectively.



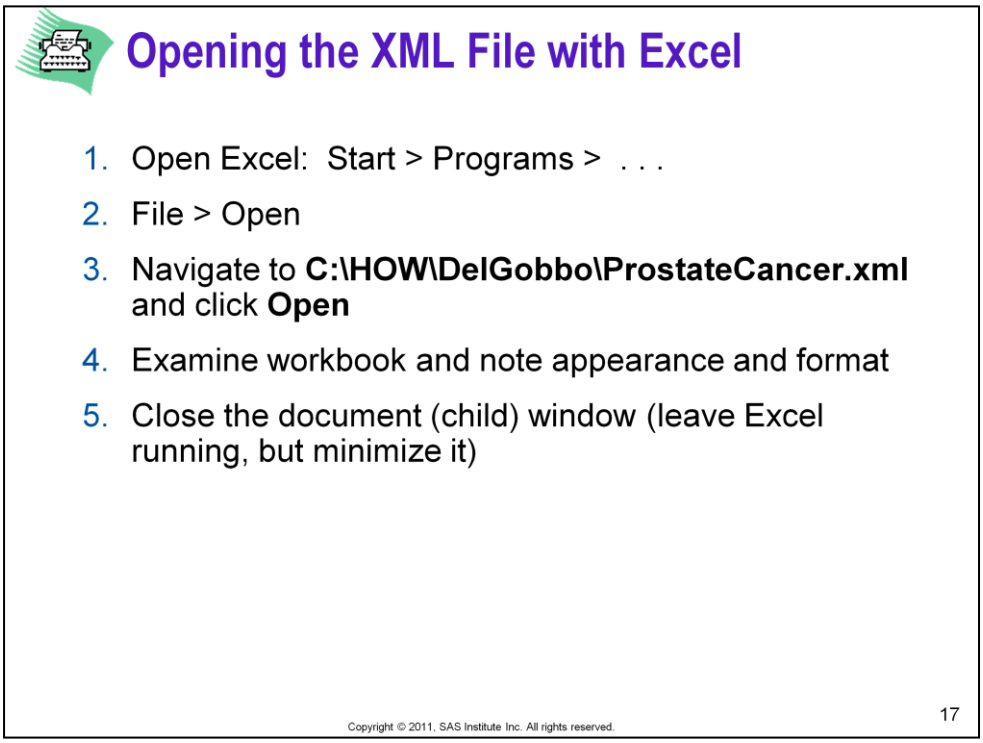
This is our first attempt to make an XML file that can be opened with Excel.

The ExcelXP tagset is used to generate XML output, and the XLsansPrinter style controls the appearance of the output.

The XML created by ODS is stored in a file named "ProstateCancer.xml", residing in the directory specified by the SAMPDIR macro variable.

By default, the ExcelXP tagset creates a new worksheet each time a SAS procedure creates new tabular output. The PROC REPORT is run with a BY statement and creates 4 worksheets, one for each distinct value of the BY variable "RX". The user-defined formats "boolean" and "rx" are applied to the "HX", "BM", and "RX" variables.

The first COMPUTE block is a placeholder for code that changes the background color of alternating rows of data. The second block 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.

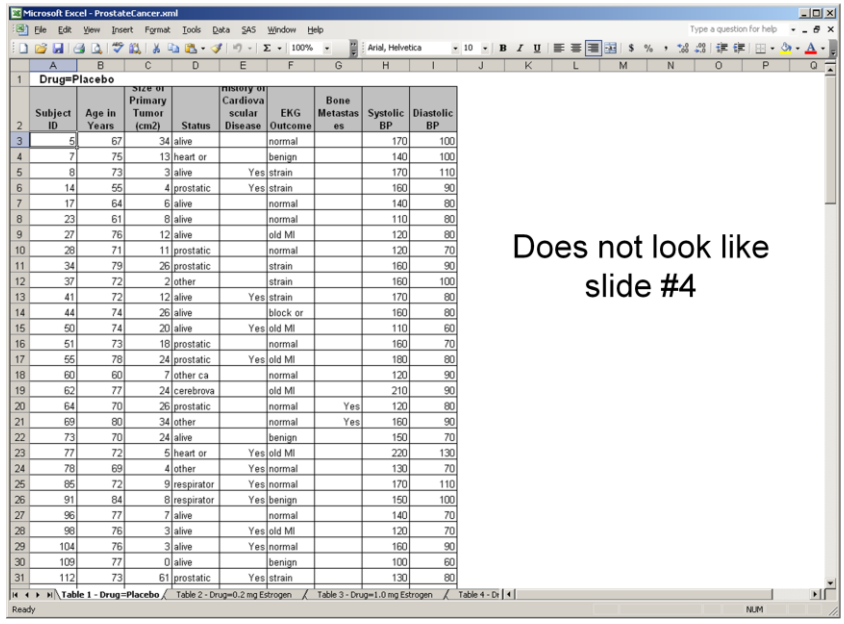


The XML file created by ODS can now be opened with Microsoft Excel.

Excel will read and convert 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**.

XML Output Viewed Using Excel

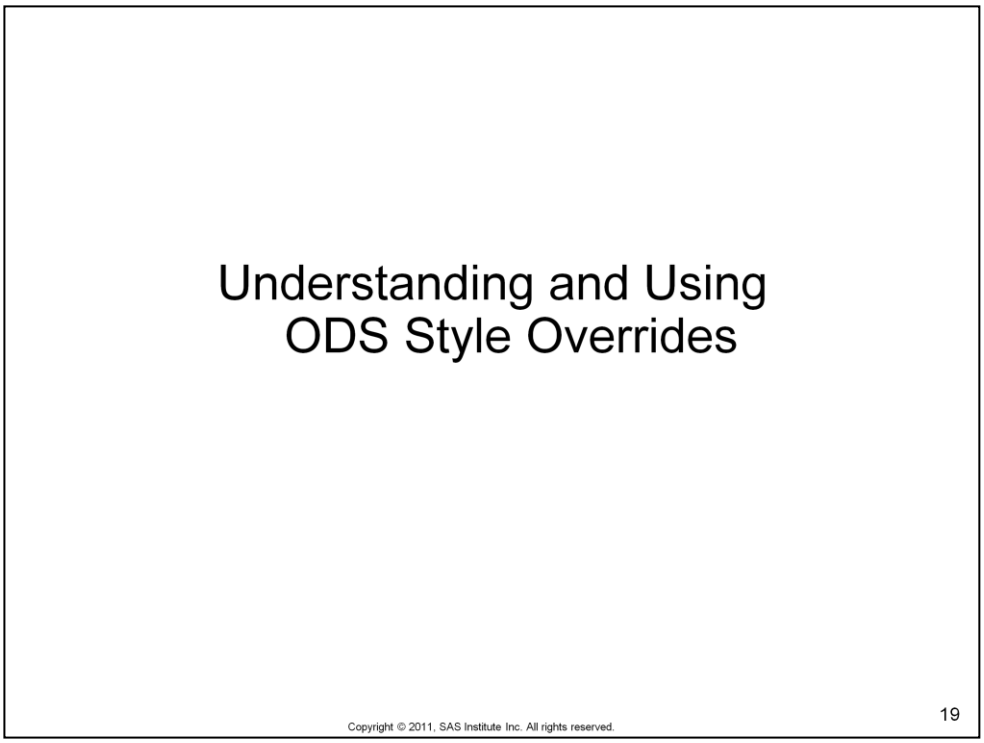


Does not look like slide #4

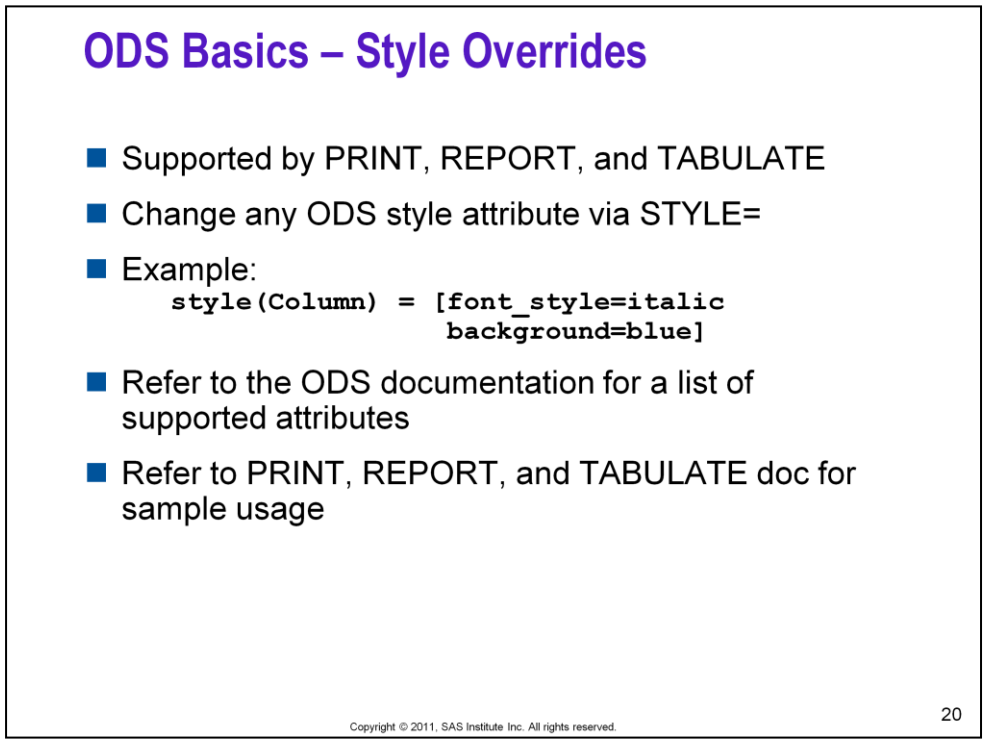
18

Problems with the output:

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 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.

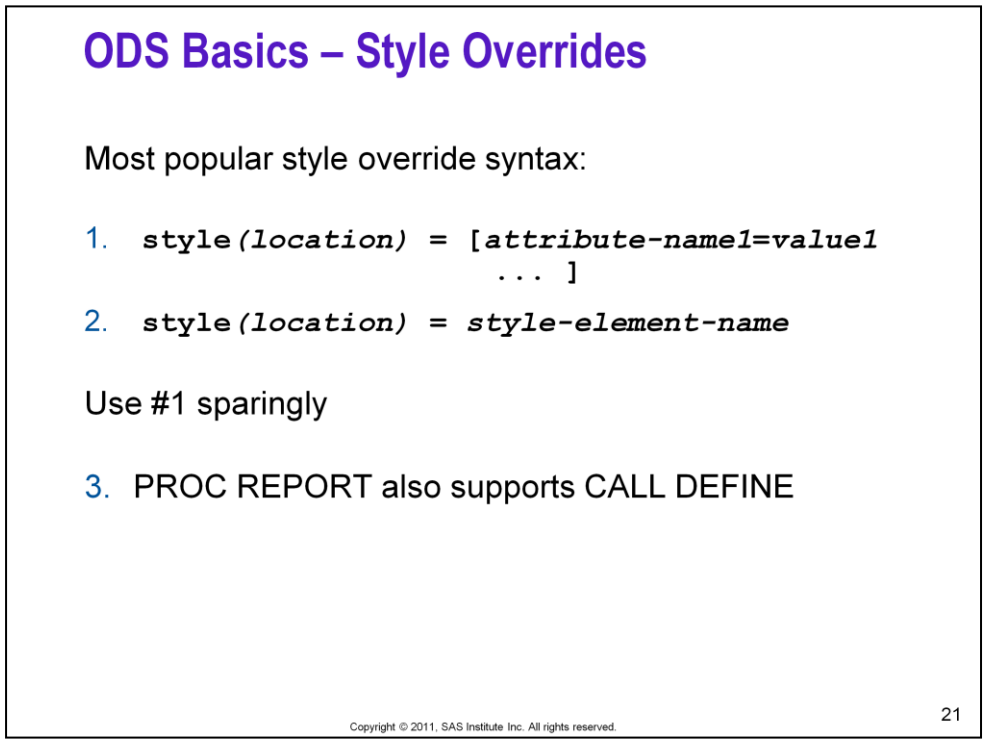


The techniques described *in this section* work only with the PRINT, REPORT, and TABULATE procedures.



ODS style overrides are used to override attributes of the ODS style you are using. They are intended to be used to make small changes to the appearance of your output, and should be used sparingly.

For example, you can use a style override to change the fonts or colors used by the XLsansPrinter style for the "Column" location of PROC REPORT, as shown here.



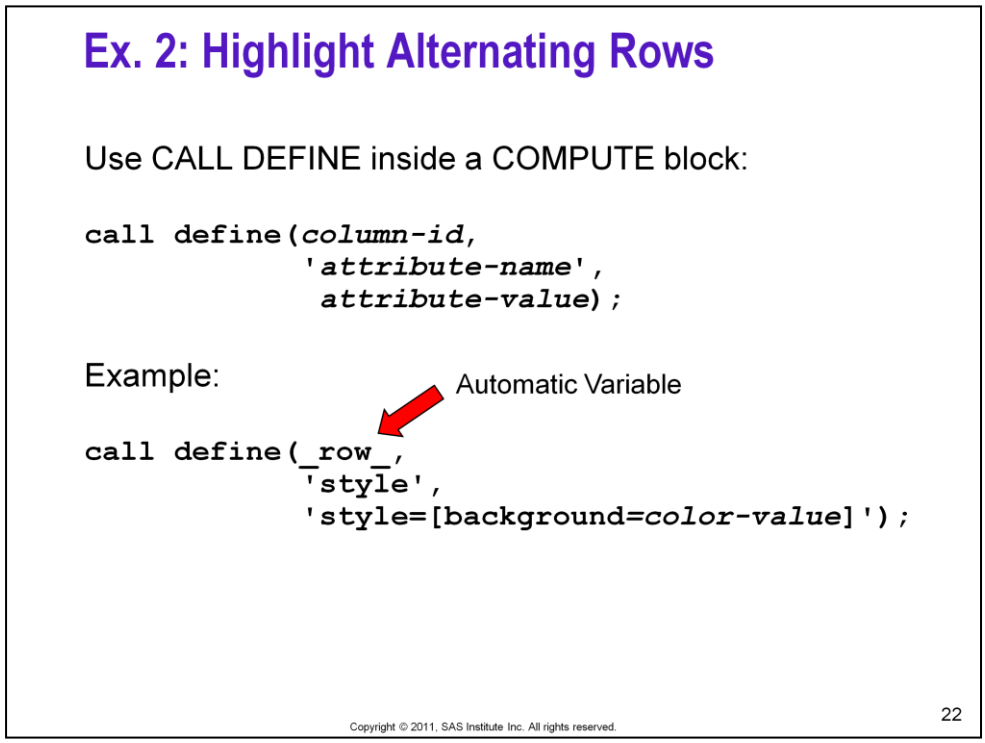
Here is the general syntax for using style overrides.

The first format uses individual style attributes defined inline, and is what you might use to change the fonts or colors defined in a style element, as shown on the previous slide. 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 overcomes these problems. 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.

The first two formats affect an entire location your output, and CALL DEFINE is used when you want to conditionally affect specific parts of your output.

Locations will be discussed later.



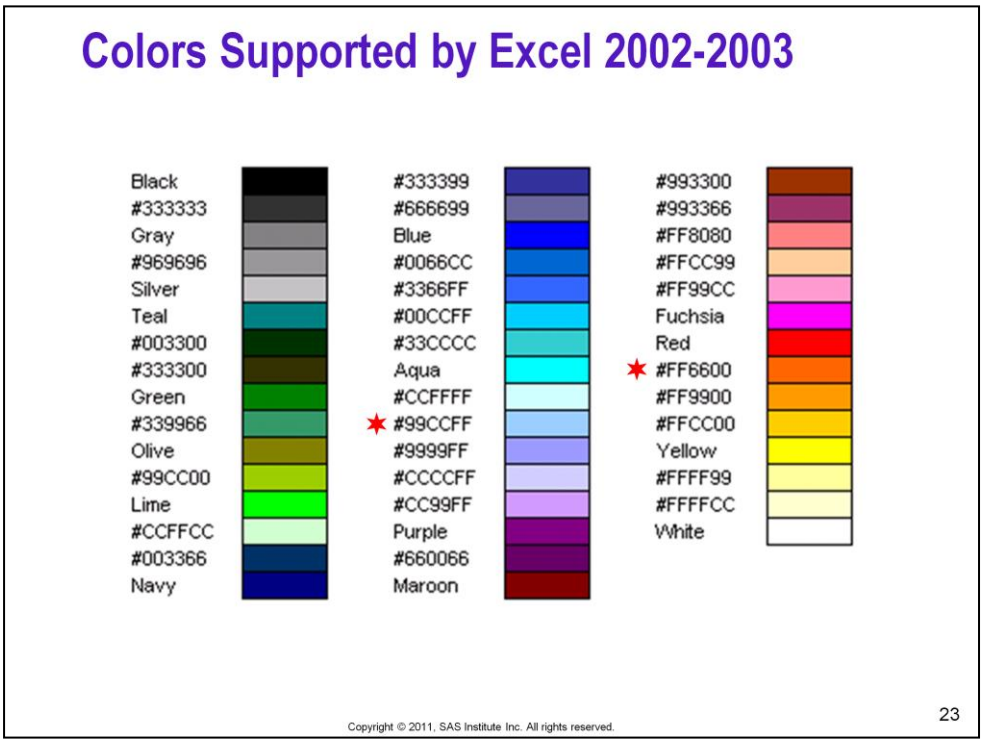
Adding a background color to alternating rows makes the worksheets more attractive and easier to read. This is accomplished by applying a style override using a CALL DEFINE statement within a COMPUTE block. The general syntax of this statement is shown above.

The CALL DEFINE statement assigns the "background" style attribute to the entire row using the automatic variable _ROW_.

References:

SAS 9.1:
http://support.sas.com/onlinedoc/913/getDoc/en/proc.hlp/a002473624.htm

SAS 9.2:
http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/a002473624.htm



Excel versions 2002 and 2003 have a limited color palette. The default colors are shown above. 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.

The colors used to highlight alternating rows and to traffic light individual cells are indicated with a star (✶)

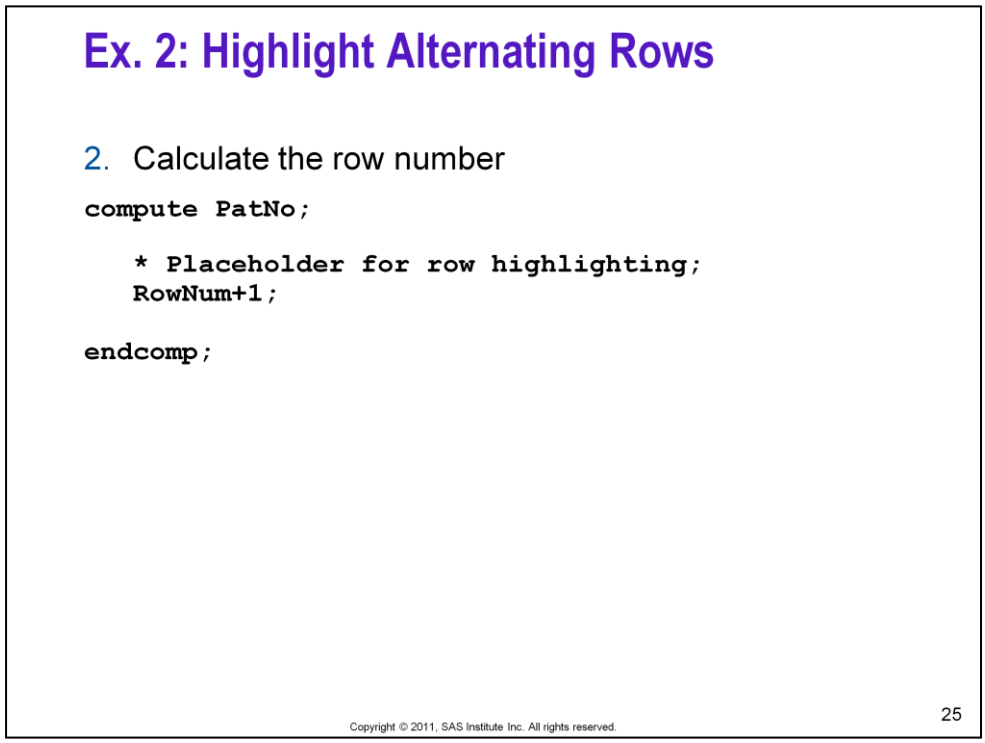
However, specifying colors is not foolproof. Because each Excel user can customize the color palette, colors are not guaranteed to exist in all instances of Excel.

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

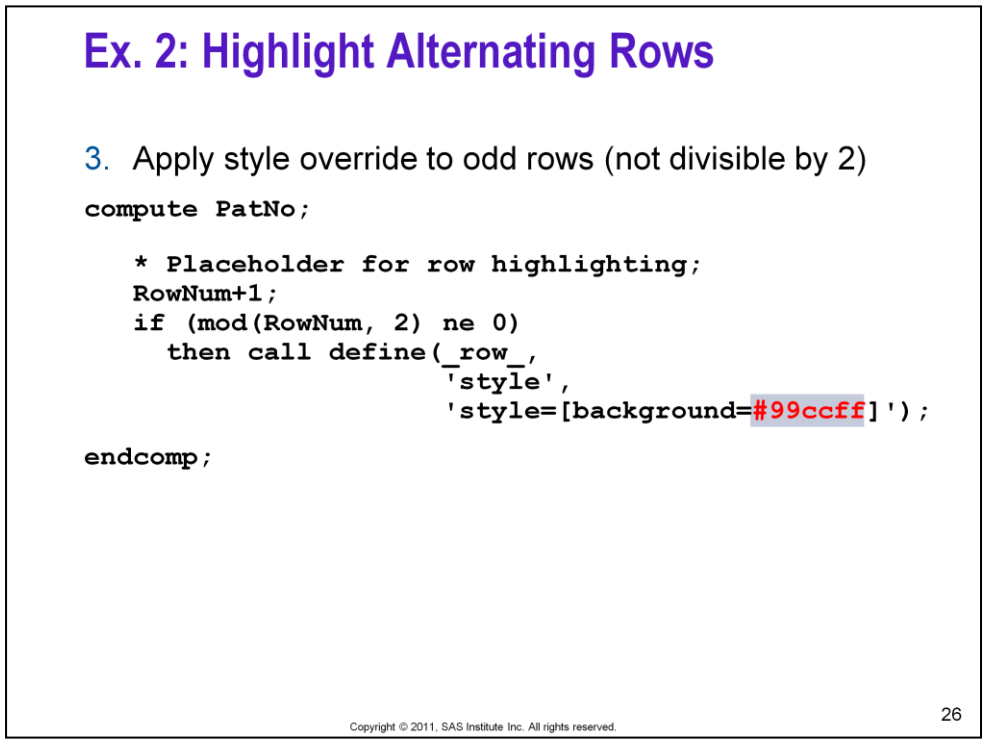
Ex. 2: Highlight Alternating Rows

1. Use COMPUTE block

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

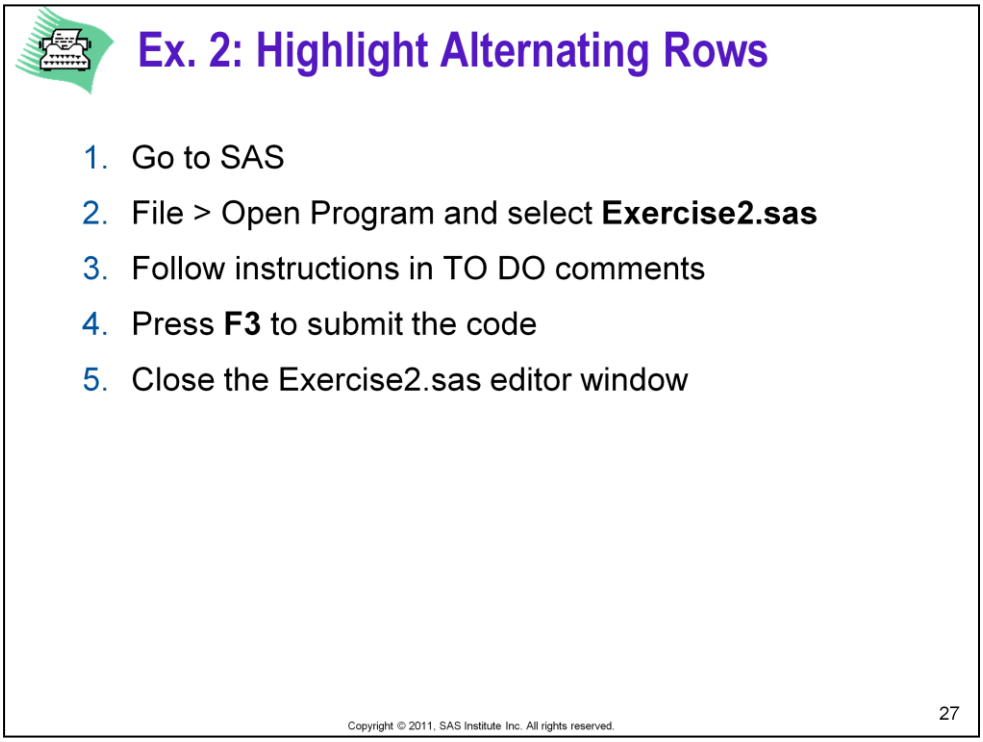


First, a new variable named "RowNum" is created in order to keep track of the row number of the data.



CALL DEFINE is most often used when you want to conditionally apply a style override to your output.

The MOD function is used to determine if the value of "RowNum" is evenly divisible by 2, and if it is not, a style override changes background color of the row to blue using the automatic variable _ROW_.



TO DO:

Line 36: Add code to change the BACKGROUND style attribute to `#99ccff`

Solution:

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

Ex. 2: Highlight Alternating Rows

1. Go to Excel
2. File > **\HOW\DelGobbo\ProstateCancer.xml**
   - or -
   ProstateCancer.xml (from the recent file list)
3. Note highlighting of alternating rows
4. Close the document (child) window (leave Excel running, but minimize it)

Ex. 3: Traffic Light Specific Cells

Have:

Subject ID	Age in Years	Size of Primary Tumor (cm2)	Status
5	67	34	alive
7	75	13	dead - heart or vascular
8	73	3	alive
14	55	4	dead - prostatic ca
17	64	6	alive

Want:

Subject ID	Age in Years	Size of Primary Tumor (cm2)	Status
5	67	34	alive
7	75	13	dead - heart or vascular
8	73	3	alive
14	55	4	dead - prostatic ca
17	64	6	alive

29

We want to draw attention to the "Status" value for patients who died as a result of heart of vascular causes.

Ex. 3: Traffic Light Specific Cells

1. Use a COMPUTE block

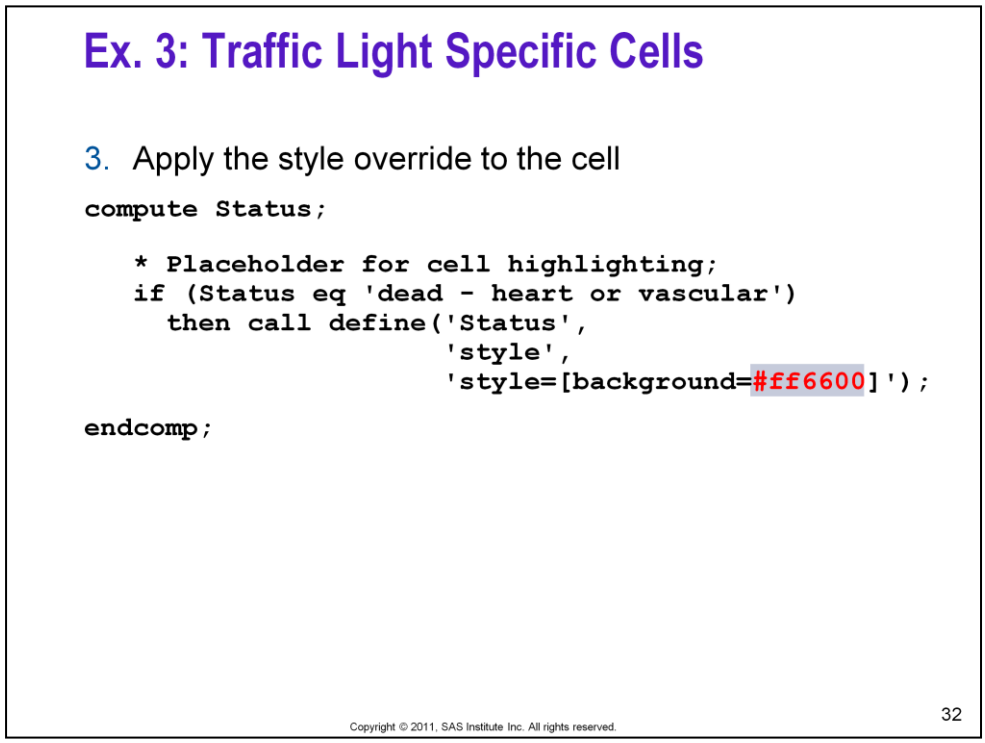
```
compute Status;

   * Placeholder for cell highlighting;

endcomp;
```

Ex. 3: Traffic Light Specific Cells

2. Specify the condition

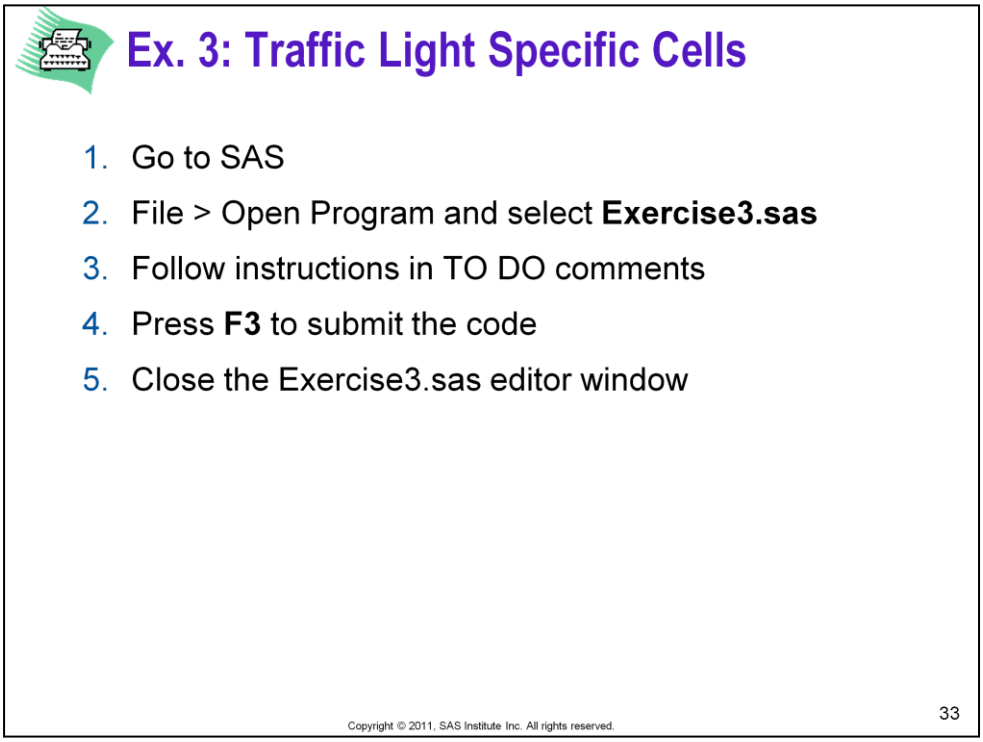
```
compute Status;

   * Placeholder for cell highlighting;
   if (Status eq 'dead - heart or vascular') ...

endcomp;
```


Next we specify the condition that must be met in order to traffic light the cell.



We use CALL DEFINE to conditionally apply a style override to our output.

We specify "Status", the name of the column to apply the style override, instead of using the automatic variable _ROW_ .



TO DO:

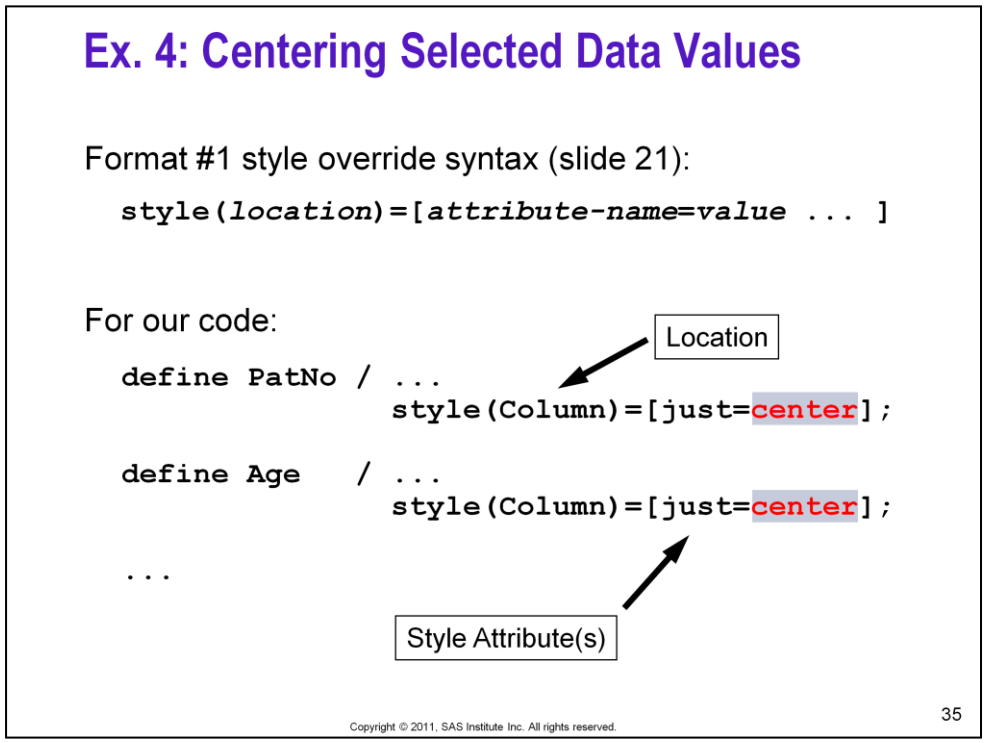
Line 43: Add code to change the BACKGROUND style attribute to `#ff6600`

Solution:

```
compute Status;
  * Placeholder for cell highlighting;
    if (Status eq 'dead - heart or vascular')
      then call define('Status', 'style',
                       'style=[background=#FF6600]');
endcomp;
```

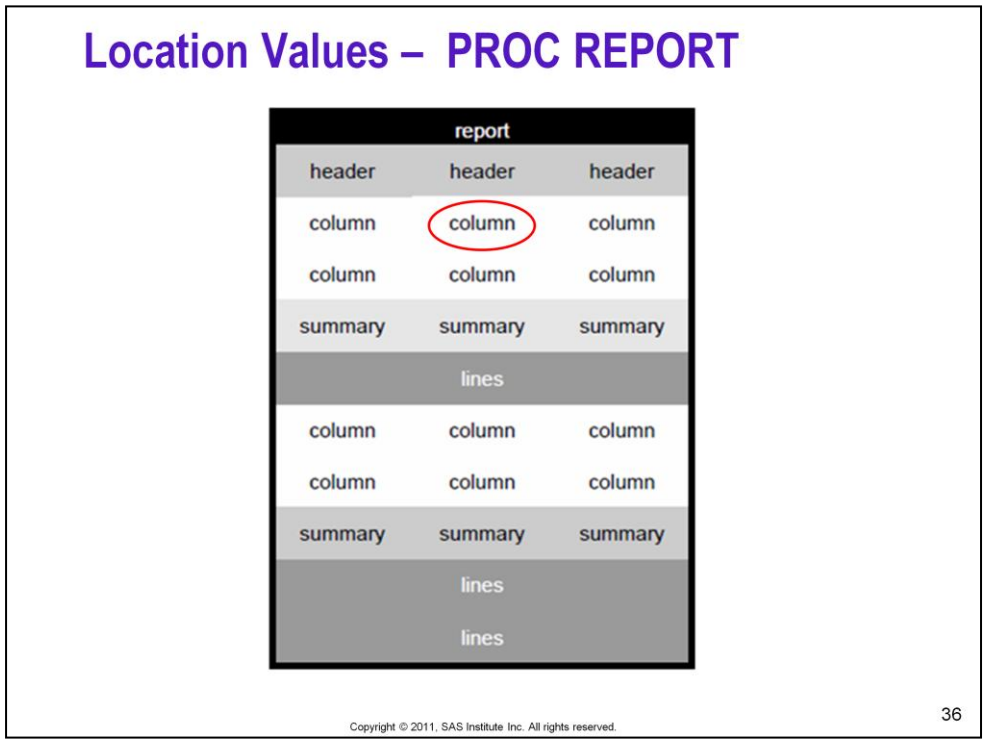
Ex. 3: Traffic Light Specific Cells

1. Go to Excel
2. File > **\HOW\DelGobbo\ProstateCancer.xml**
   - or -
   ProstateCancer.xml (from the recent file list)
3. Note traffic lighting
4. Close the document (child) window (leave Excel running, but minimize it)



We have used CALL DEFINE to conditionally apply style overrides to specific parts of our output. Now we want to center *all* of the data values in every column except "Status" and "EKG". This is accomplished by adding the STYLE option to the existing DEFINE statements.

Note that we are using format #1 from slide 21 (individual style attributes specified inline).

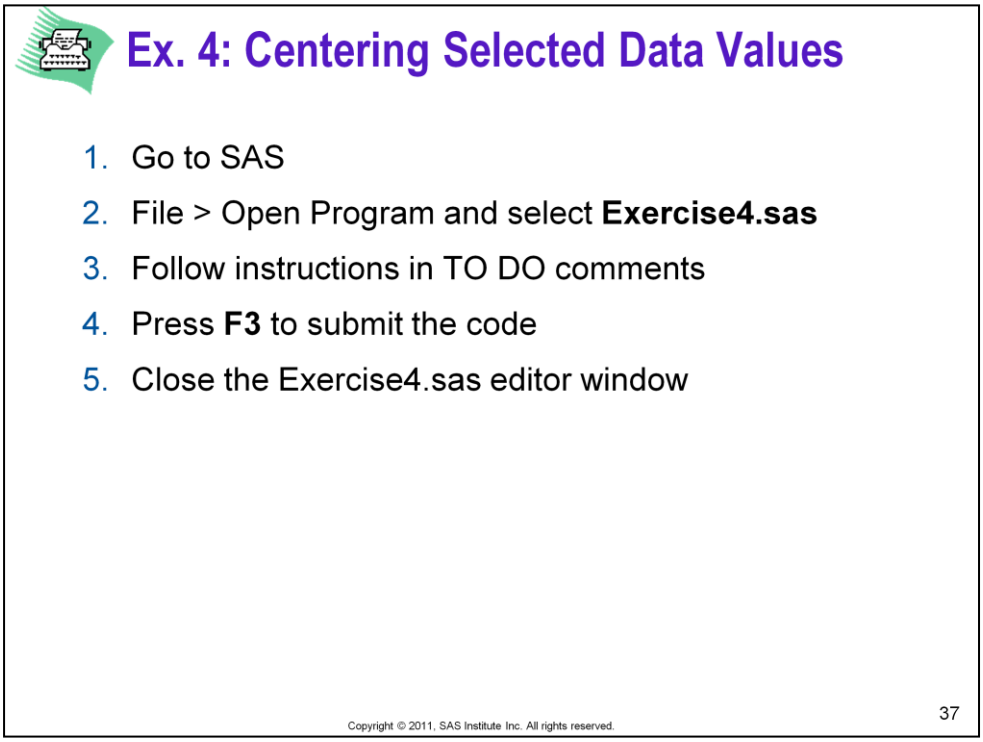


Style overrides are used to apply style attributes or elements to specific parts of SAS output called *locations*. Here you can see the locations that are pertinent to the PROC REPORT output.

To change the appearance of the data cells in our output, we apply style overrides to the "Column" location.

Reference:

http://support.sas.com/rnd/base/ods/scratch/reporting-styles-tips.pdf



TO DO:

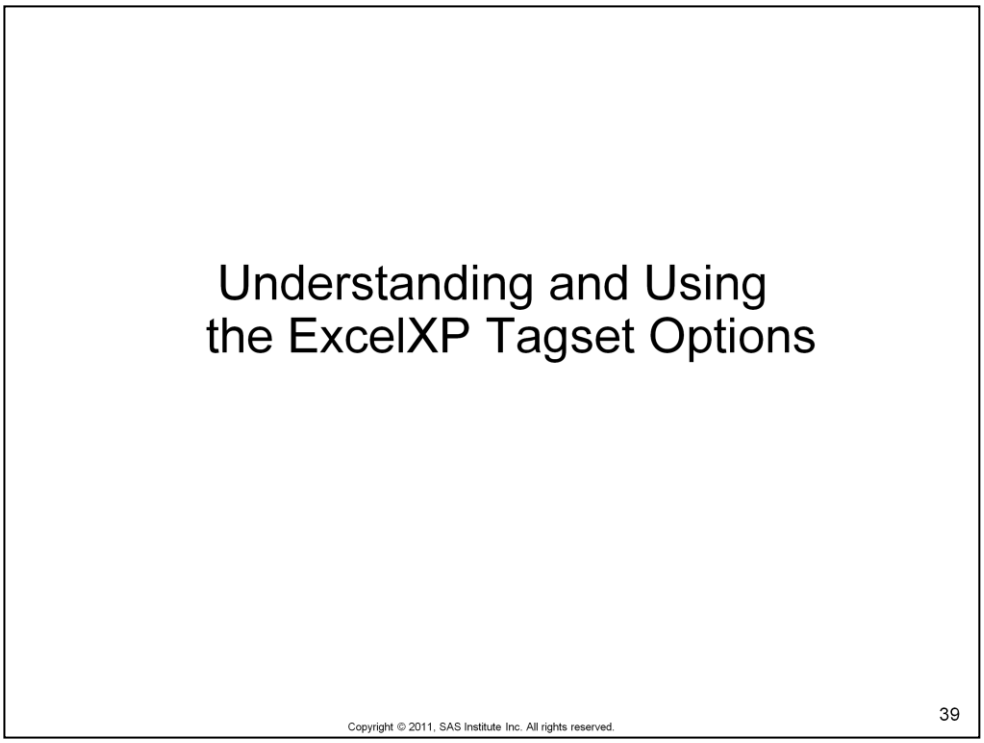
Lines 22, 23, 24, 26, 27, 28, 29, and 30: Add code to center text

Solution:

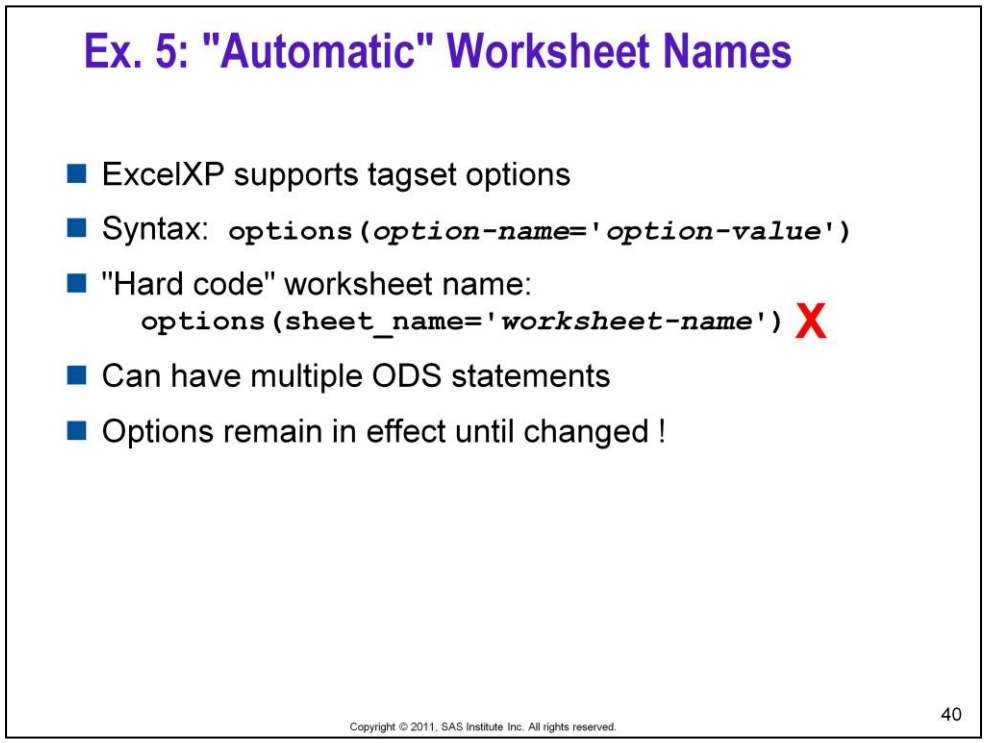
```
style(Column)=[just=center]
```

Ex. 4: Centering Selected Data Values

1. Go to Excel
2. File > **\HOW\DelGobbo\ProstateCancer.xml**
   - or -
   ProstateCancer.xml (from the recent file list)
3. Note centered data values
4. Close the document (child) window (leave Excel running, but minimize it)



Unlike style overrides discussed in the previous section, *the following techniques apply to all SAS procedure output*.

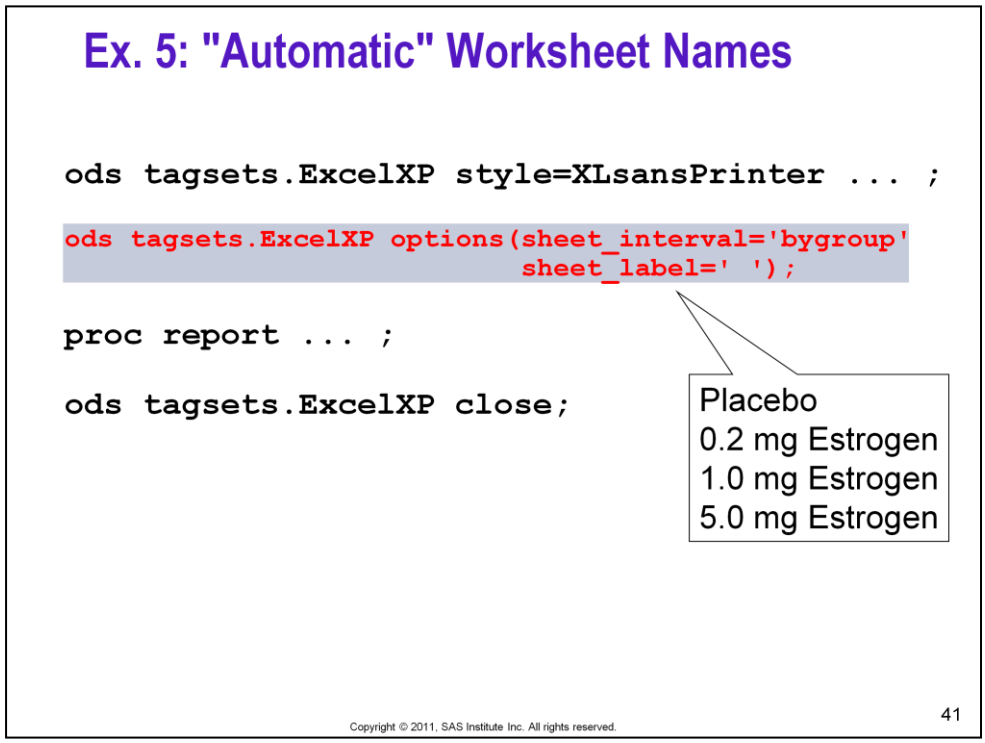


The ExcelXP tagset supports many options that control both the appearance and functionality of the Excel workbook. Many of these tagset options are simply tied straight into existing Excel options or features. Tagset options are specified in an ODS statement using the OPTIONS keyword, as shown above.

ODS generates a unique name for each worksheet, as required by Microsoft Excel. These names are generally unappealing. There are, however, several tagset options that you can use to alter the names of the worksheets.

Use the SHEET_NAME option to explicitly specify a worksheet name. But instead of specifying hard-coded worksheet names, we want the worksheets to be named automatically, based on the value of the BY variable "RX".

IMPORTANT NOTE: Tagset options remain in effect until they are changed !



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 as shown here, the current value of the first BY group variable, without any prefix, is used as the worksheet name.

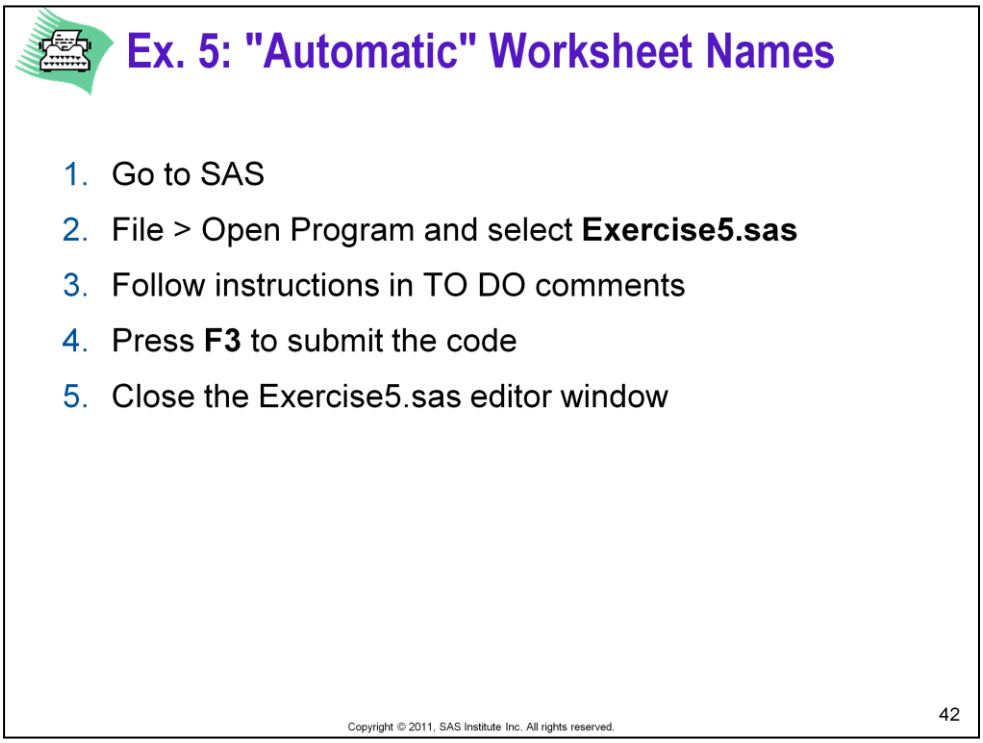
The BY variable "RX" in our REPORT procedure code contains 4 distinct values, and the formatted values are used for worksheet names:

Placebo

0.2 mg Estrogen

1.0 mg Estrogen

5.0 mg Estrogen



TO DO:

Line 15: Specify option value to cause new worksheets to be created for each BY group

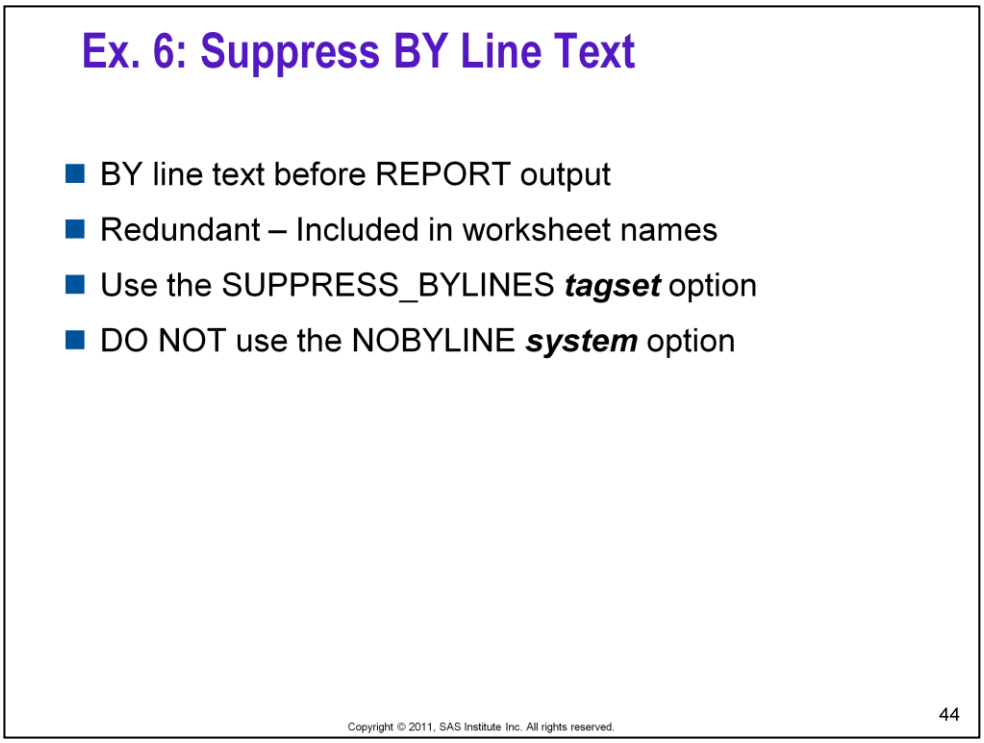
Line 16: Note the special value for SHEET_LABEL

Solution:

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

Ex. 5: "Automatic" Worksheet Names

1. Go to Excel
2. File > **\HOW\DelGobbo\ProstateCancer.xml**
   - or -
   ProstateCancer.xml (from the recent file list)
3. Note worksheet names
4. Close the document (child) window (leave Excel running, but minimize it)



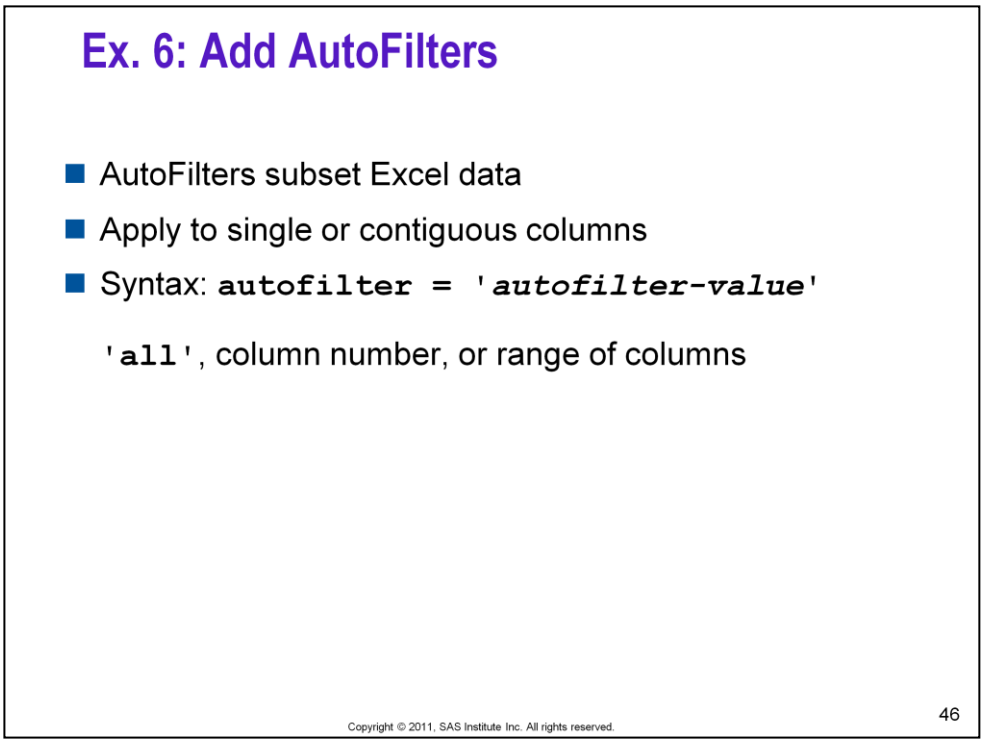
BY line text appears in the worksheets because the REPORT procedure is executed with a BY statement. The text is redundant because the BY value is displayed in the worksheet names. To omit the BY line text, specify the SUPPRESS_BYLINES option in the ODS statement that precedes the PROC REPORT code.

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

Ex. 6: Suppress BY Line Text

```
ods tagsets.ExcelXP style=XLsansPrinter ... ;

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

proc report ... ;

ods tagsets.ExcelXP close;
```

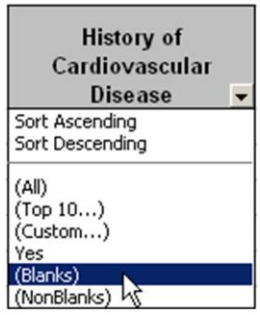

The SUPPRESS_BYLINES option is used to prevent the BY line text from being included in the worksheets.



An Excel AutoFilter allows you to filter, or subset, the data that is being displayed in a worksheet.

Ex. 6: Add AutoFilters

Example: Patients without cardiovascular disease



47

A column of data containing an AutoFilter is indicated by an arrow button in the header cell of that column.

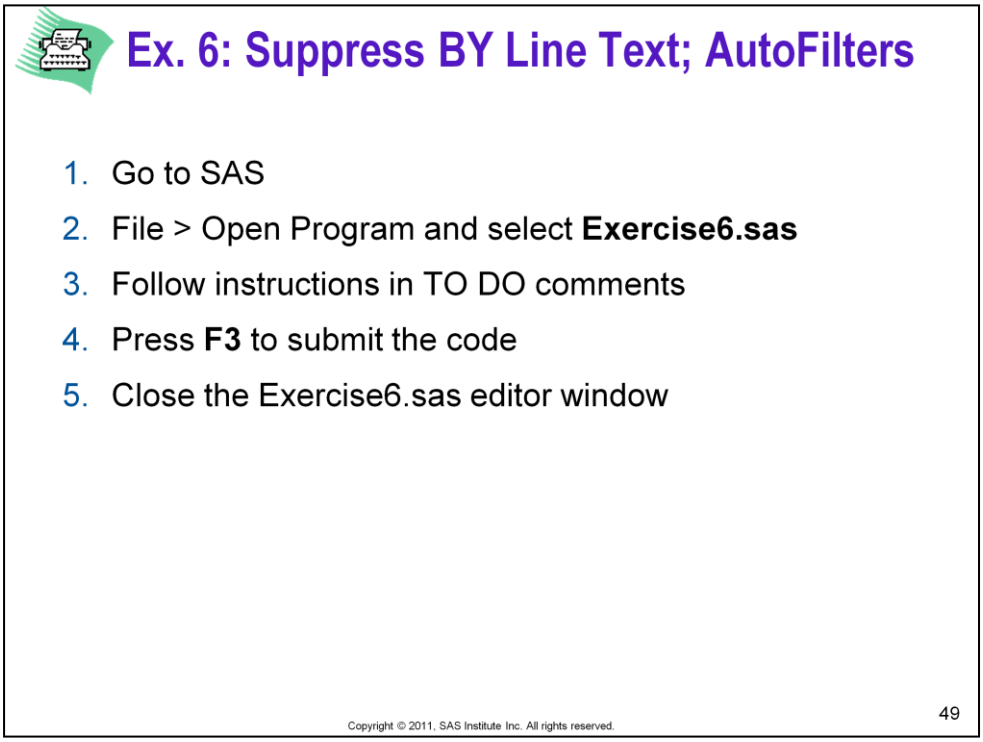
Suppose you want to view only records for patients that do not have a history of cardiovascular disease. You click the AutoFilter button on the "History of Cardiovascular Disease" column, and then select "(Blanks)".

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.

Ex. 6: Add AutoFilters

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

We want AutoFilters on columns B through G, so we specify the value "2–7" for the AUTOFILTER option.



TO DO:

Line 16: Specify the option value to suppress BY lines

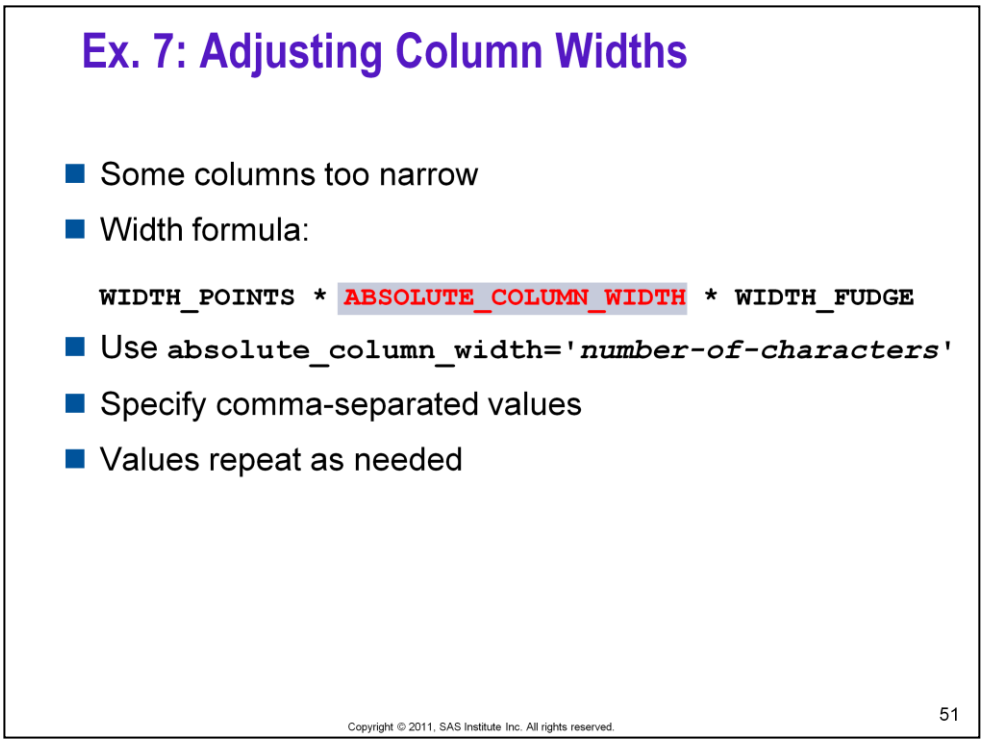
Line 17: Specify the option to add AutoFilters to columns 2-7

Solution:

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

Ex. 6: Suppress BY Line Text; AutoFilters

1. Go to Excel
2. File > **\HOW\DelGobbo\ProstateCancer.xml**
   - or -
   ProstateCancer.xml (from the recent file list)
3. No BY values in worksheet; AutoFilters
4. Close the document (child) window (leave Excel running, but minimize it)



Some of the column widths of the lab results worksheets are too narrow, causing data to be obscured, or unattractive wrapping of column headings. The ABSOLUTE_COLUMN_WIDTH option is used to correct this problem.

The ExcelXP tagset approximates column widths, using the formula above, sometimes resulting in less than perfect widths.

The ABSOLUTE_COLUMN_WIDTH option accepts a list of comma-separated values that specify the width, in characters, of the columns. If you specify fewer values than the number of columns in the worksheet, the values you specify will repeat, and are used for the remaining columns.

Ex. 7: Adjusting Column Widths

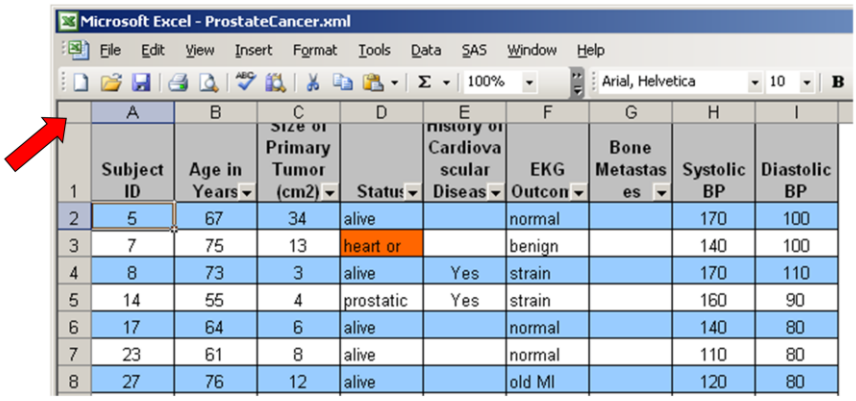
Examples, assuming 7 columns of data

ABSOLUTE_COLUMN_WIDTH	Column Widths
10	10,10,10,10,10,10,10
5,10	5,10,5,10,5,10,5
5,10,15	5,10,15,5,10,15,5

These examples show how the specified value(s) repeat where there are more columns than option values.

Ex. 7: Adjusting Column Widths

1. Select all columns

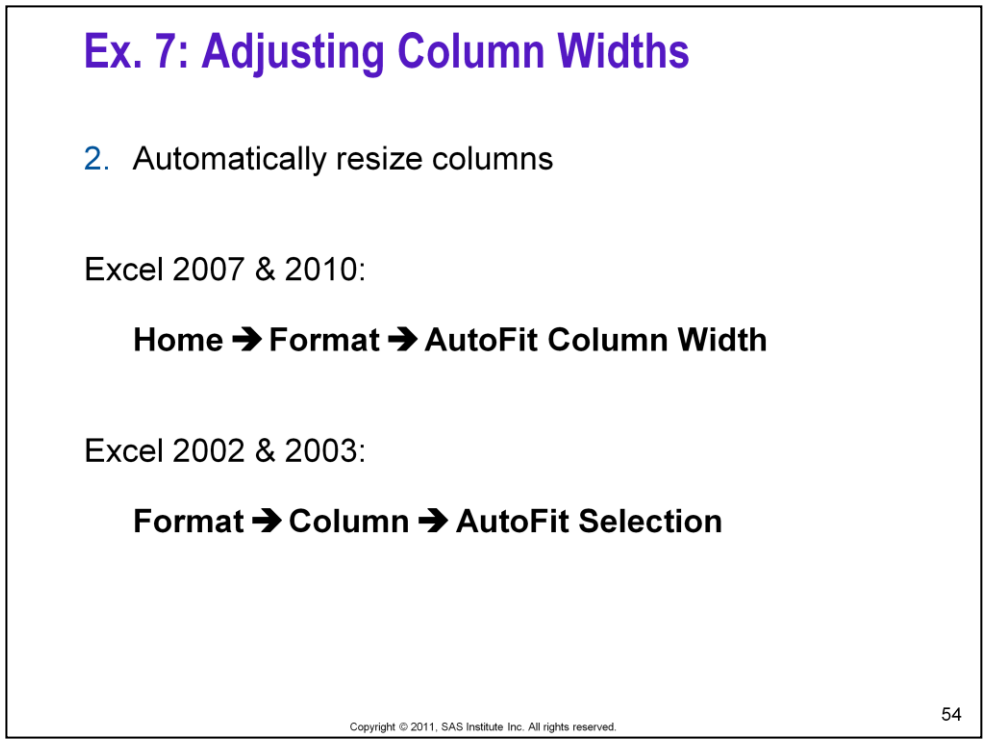


	A	B	C	D	E	F	G	H	I
1	Subject ID	Age in Years	Size of Primary Tumor (cm2)	Status	History of Cardiovascular Diseas	EKG Outcon	Bone Metastases	Systolic BP	Diastolic BP
2	5	67	34	alive		normal		170	100
3	7	75	13	heart or		benign		140	100
4	8	73	3	alive	Yes	strain		170	110
5	14	55	4	prostatic	Yes	strain		160	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

53

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.

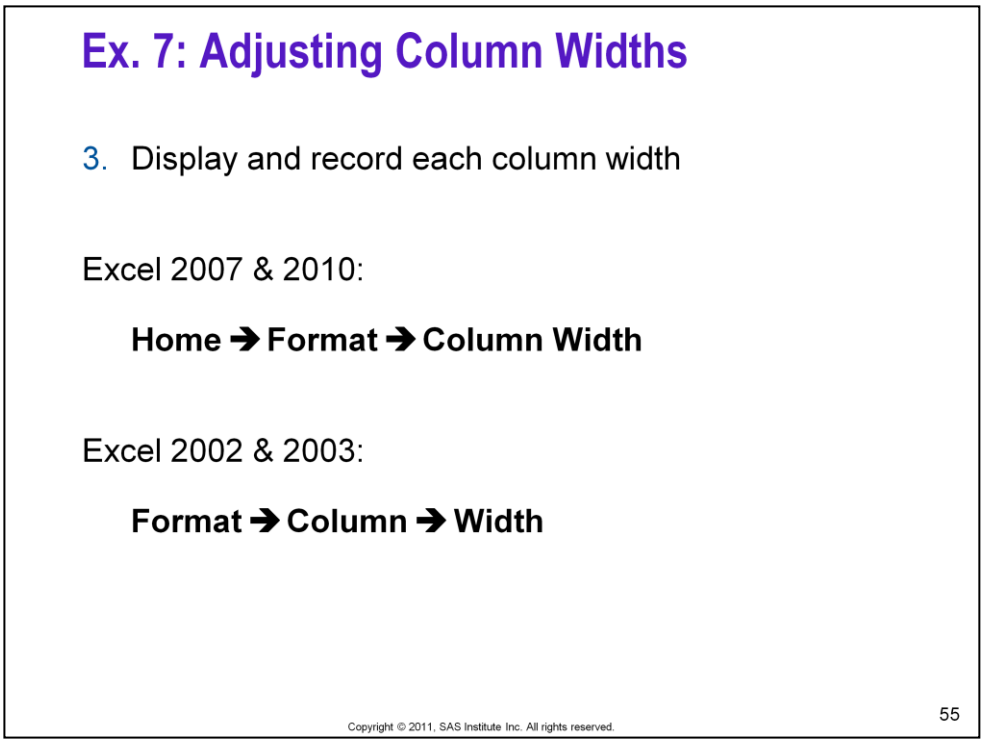


Next, click:

Excel 2007 and 2010: **Home ➔Format ➔AutoFit Column Width**

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.



To display the values to use for ABSOLUTE_COLUMN_WIDTH, select a cell in a column and then click:

Excel 2007 and 2010: **Home ➔Format ➔Column Width**

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.

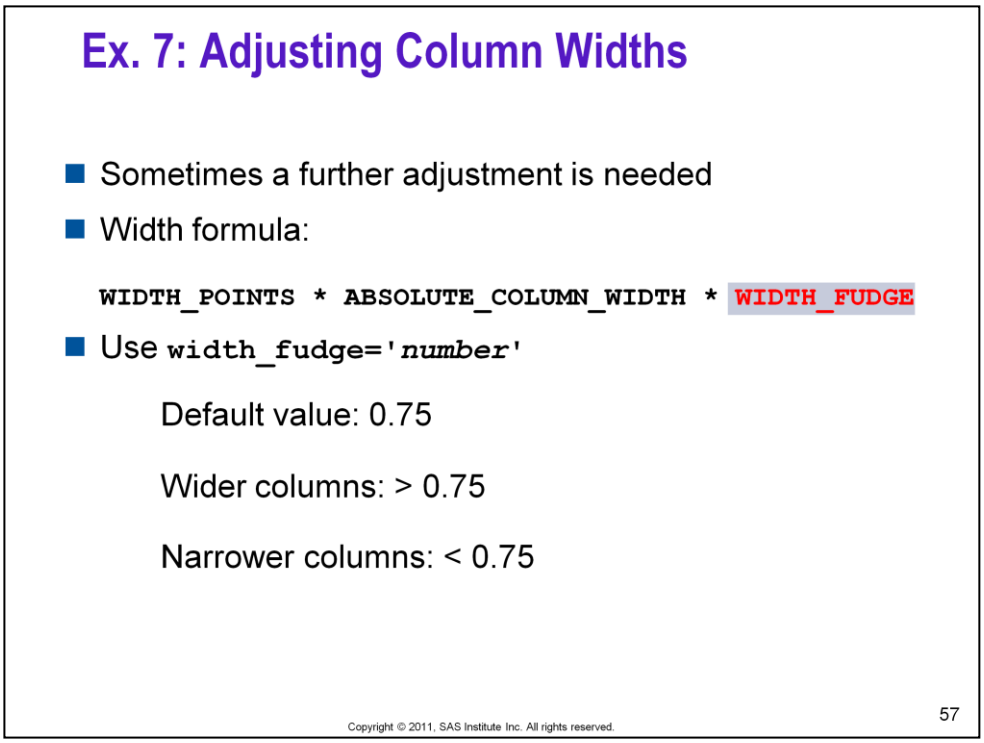
Ex. 7: Adjusting Column Widths

4. Specify values for ABSOLUTE_COLUMN_WIDTH

```
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');
```

(ignore line wrapping above)

Copyright © 2011, SAS Institute Inc. All rights reserved.



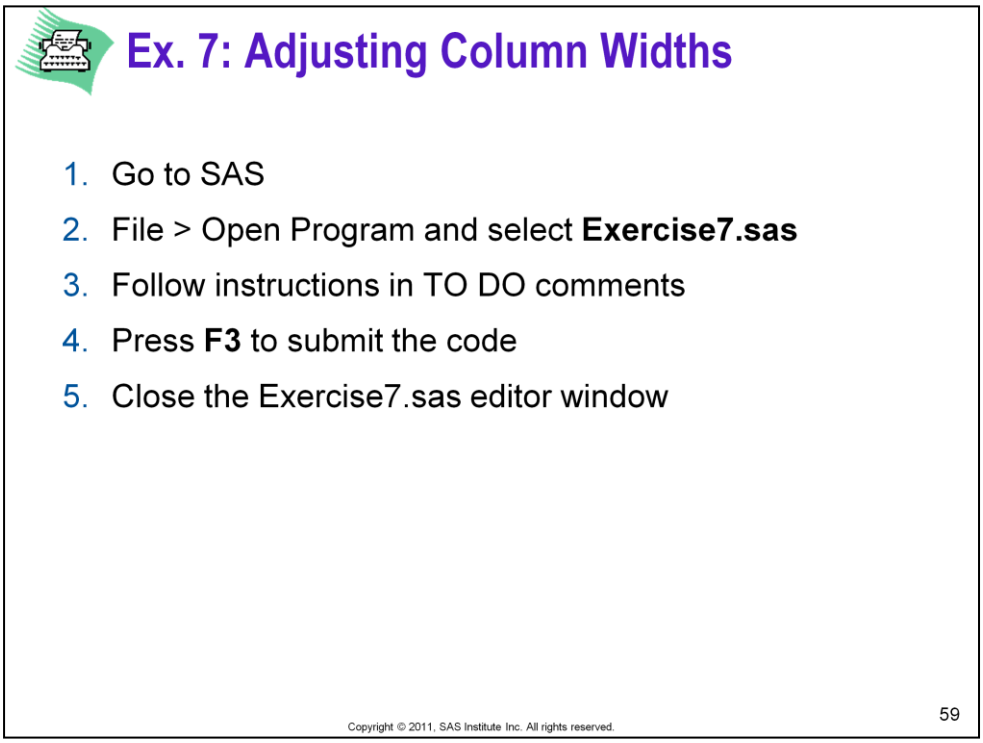
If the tagset computes widths that are still not satisfactory, you can make small adjustments to *all* columns widths by using the WIDTH_FUDGE option. For our data, a value of 0.58 produces more appropriate column widths than the default value of 0.75.

Ex. 7: Adjusting Column Widths

5. Specify value for WIDTH_FUDGE

```
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');
```

(ignore line wrapping above)



TO DO:

Line 18: Specify the 9 values for ABSOLUTE_COLUMN_WIDTH

Line 19: Note the value for WIDTH_FUDGE

Solution:

```
ods tagsets.ExcelXP options( ...
absolute_column_width='7.14,10.43,18.29,23.57,18.29,27.71,14.43,7.29,8'
width_fudge='0.58');
```

Ex. 7: Adjusting Column Widths

1. Go to Excel
2. File > **\HOW\DelGobbo\ProstateCancer.xml**
   - or -
   ProstateCancer.xml (from the recent file list)
3. Note new column widths and heading heights
4. Close the document (child) window (leave Excel running, but minimize it)

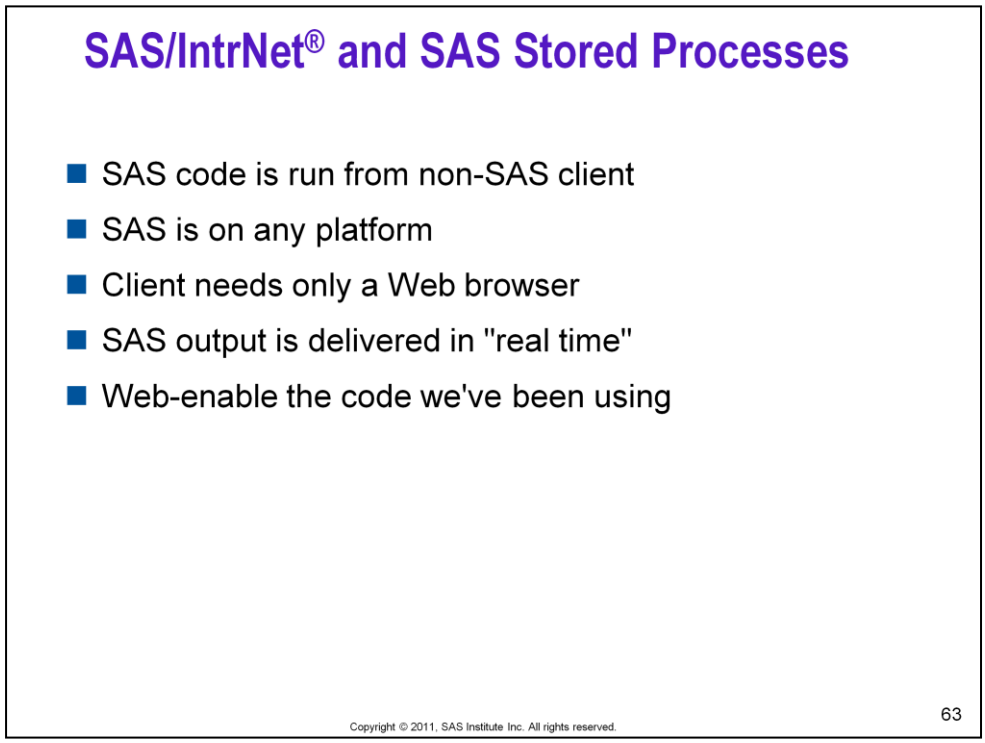
Summary: Creating Multi-Sheet Workbooks

- Use ExcelXP tagset to create XML file
- Resulting XML file can be viewed with Excel
- Apply ODS style overrides carefully
- Make use of tagset options
- Use Excel formats instead of SAS formats

Using SAS/IntrNet®

and

SAS Stored Processes



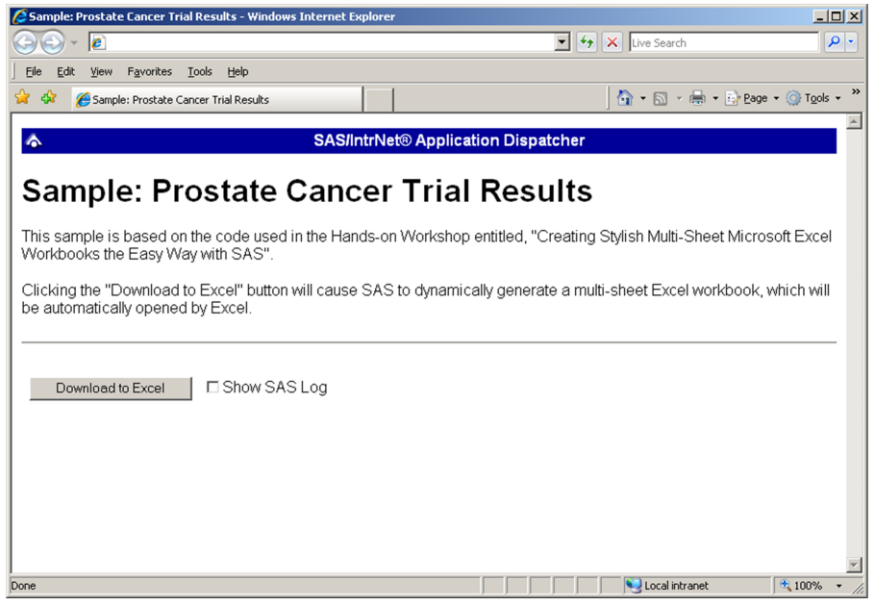
The purpose of the SAS/IntrNet® Application Dispatcher or SAS Stored Processes are to allow you to execute SAS programs from a client machine that does not have SAS installed. The client machine *may* have SAS installed, but that is not required.

A typical client-server model is followed. The SAS server can reside on any hardware platform (Windows, UNIX, z/OS, etc.) and is standing by, waiting to execute a SAS program. The most common client is a Web browser, again, running on any platform.

When the "OK" button of the Web page is clicked, input parameters, if any, are sent to the SAS server. Your SAS code executes, and the output is delivered in "real time" to the Web browser.

The following slides illustrate this process, using a "Web-enabled" version of the SAS code we have been working with.

Dynamically-Generated XML



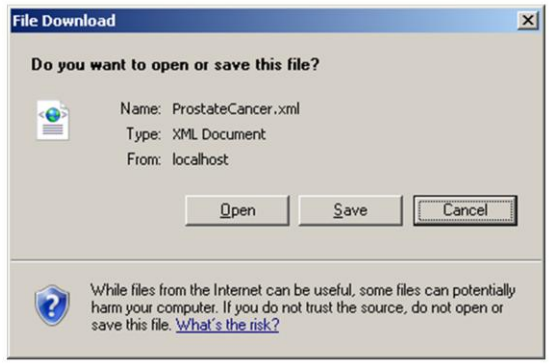
64

Here is a simple Web page that is used to execute SAS code stored on a server using the SAS/IntrNet® Application Dispatcher.

The code that is executed is substantially similar to the final version of the code that we used to generate XML file, with a few changes to "Web-enable" it.

Clicking "Download to Excel" causes the SAS program to execute on the server.

Dynamically-Generated XML



65

Once the SAS program executes, the results are sent back to the Web browser.

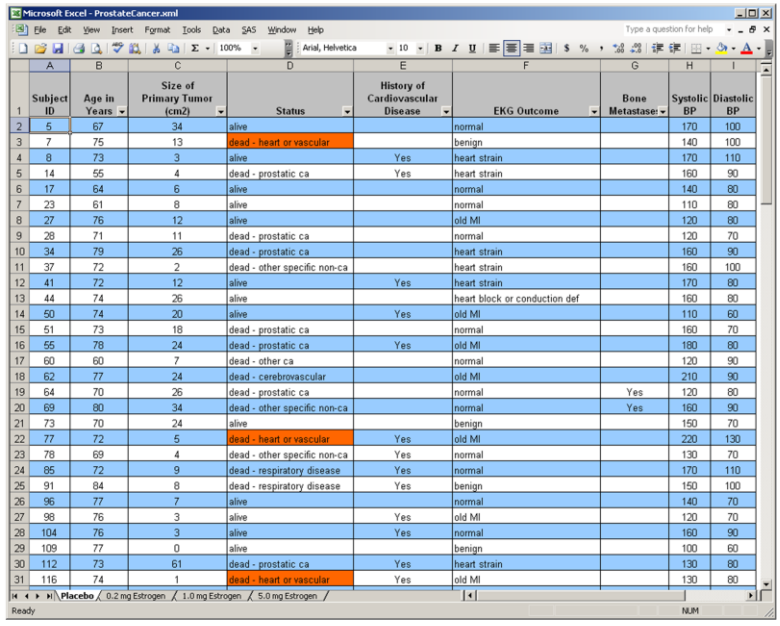
Note that you are presented with a "File Download" dialog box, instead of the results being displayed in the Web browser. Clicking "Open" causes the SAS output to be displayed in Excel, provided that Excel is installed on the client machine.

This code, specific to SAS/IntrNet®, must be specified before any ODS statements. It causes the SAS output to be displayed in Excel instead of the Web browser:

```
%let RV =%sysfunc(appsrv_header(Content-type,application/vnd.ms-excel));
```

This code will also work with SAS Stored Processes.

Dynamically-Generated XML



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

66

Here is the SAS output, created in "real time", and delivered to the client. Note that Excel is used to view the SAS output, not the Web browser.

Refer to the accompanying paper for more information about this topic.

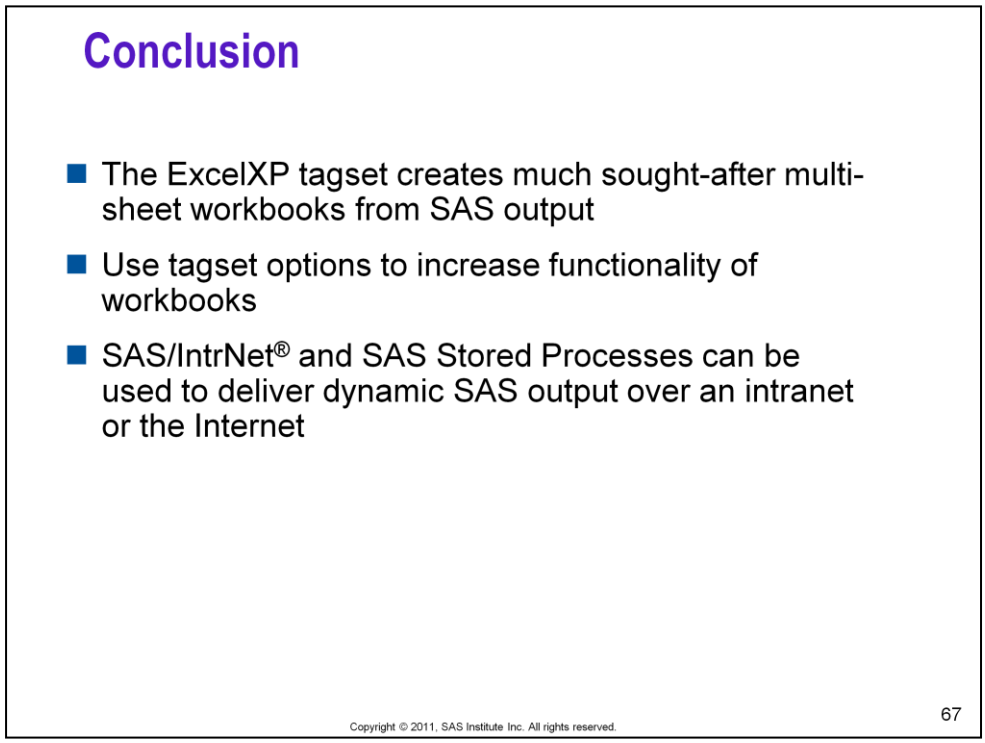
References:

SAS/IntrNet® Application Dispatcher 9.1:
http://support.sas.com/onlinedoc/913/docMainpage.jsp?_topic=dispatch.hlp/main_contents.htm

SAS/IntrNet® Application Dispatcher 9.2:
http://support.sas.com/documentation/cdl/en/dispatch/59547/HTML/default/main_contents.htm

SAS Stored Processes 9.1:
http://support.sas.com/rnd/itech/doc9/dev_guide/stprocess/index.html

SAS Stored Processes 9.2:
http://support.sas.com/documentation/cdl/en/stpug/61271/HTML/default/titlepage.htm



Using ODS to generate specially-formatted XML output is an effective method of incorporating SAS output in Excel documents.

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.

SAS Institute continues to work toward better Microsoft Office integration, and future releases of SAS software will provide even more robust means of using SAS content with Microsoft Office applications.

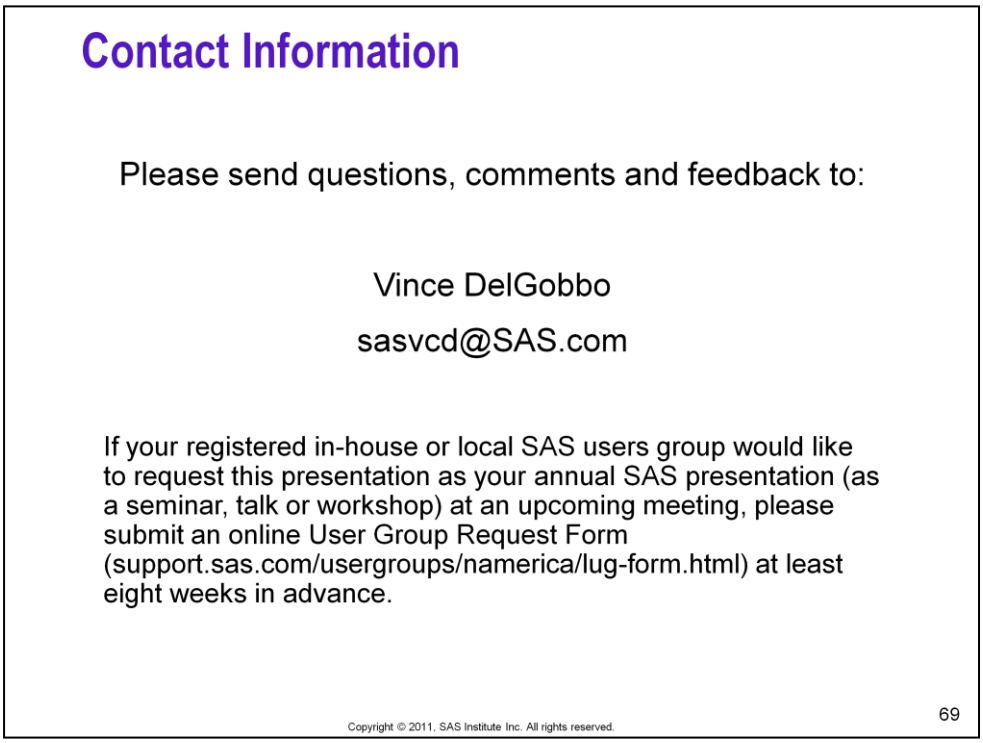
References and Further Reading

- Paper and Sample Code ("download package"):
  support.sas.com/rnd/papers/index.html#excel2011
- ExcelXP Papers and Resources:
  www.sas.com/reg/gen/corp/867226?page=Resources

The sample programs and data used in this workshop as well as a copy of the accompanying paper are available at the SAS Presents Web site. Go to http://support.sas.com/rnd/papers/index.html#excel2011 and find the entry "Creating Stylish Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS".

IMPORTANT NOTE: Be sure to install a recent version of the ExcelXP tagset on your system. See the accompanying paper for details and instructions.

Review the "Installation" and "Usage" sections of the file named "ReadMe.txt" in the ZIP archive for important information on using the sample files.



About the author:

Vince DelGobbo is a Senior Systems Developer in the Web Tools group at SAS. This group is responsible for developing the SAS/IntrNet Application Dispatcher and SAS Stored Processes. He is the developer of the HTML Formatting Tools and the SAS Design-Time Controls, and is developing other new Web- and server-based technologies, as well as integrating SAS output with Microsoft Office. He is also involved in the development of the ExcelXP ODS tagset. Vince has been a SAS Software user since 1982, and joined SAS in 1992.