SAS Global Forum 2010

April 11 – 14, 2010

Washington State Convention and Trade Center

Seattle, WA

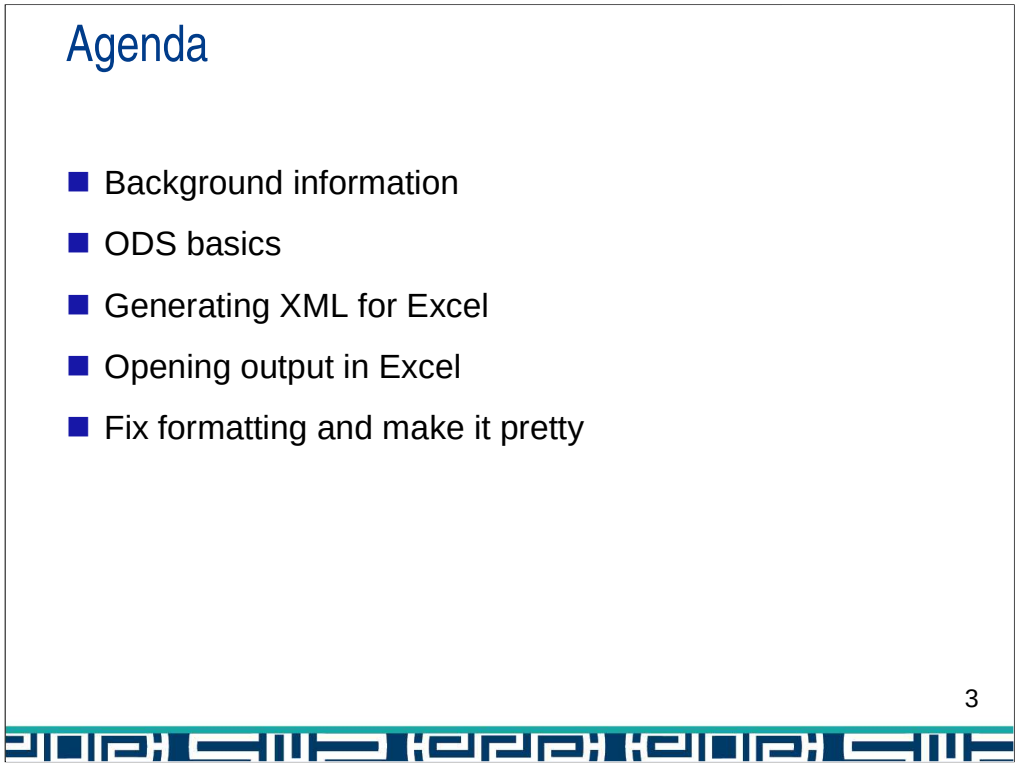
A special "thank you" to Nancy Brucken and Zul Habib for inviting me to present this topic, and to the following individuals for their valuable contributions to the accompanying paper:

- Chris Barrett of SAS Institute Inc.
- Richard Allen of Peak Statistical Services
- Nancy Brucken of i3 Statprobe
- Susan Fehrer of BioClin Inc.
- John King of Ouachita Clinical Data Services Inc.
- Alissa Ruelle of PharmaNet Inc.

Goals

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

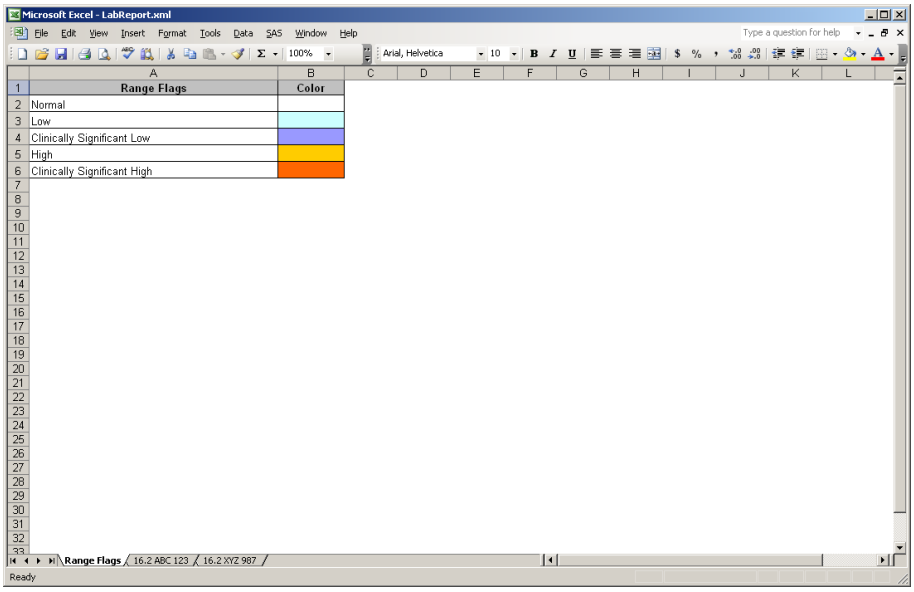
2



Software Requirements

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

PROC PRINT Output ⟶ Excel



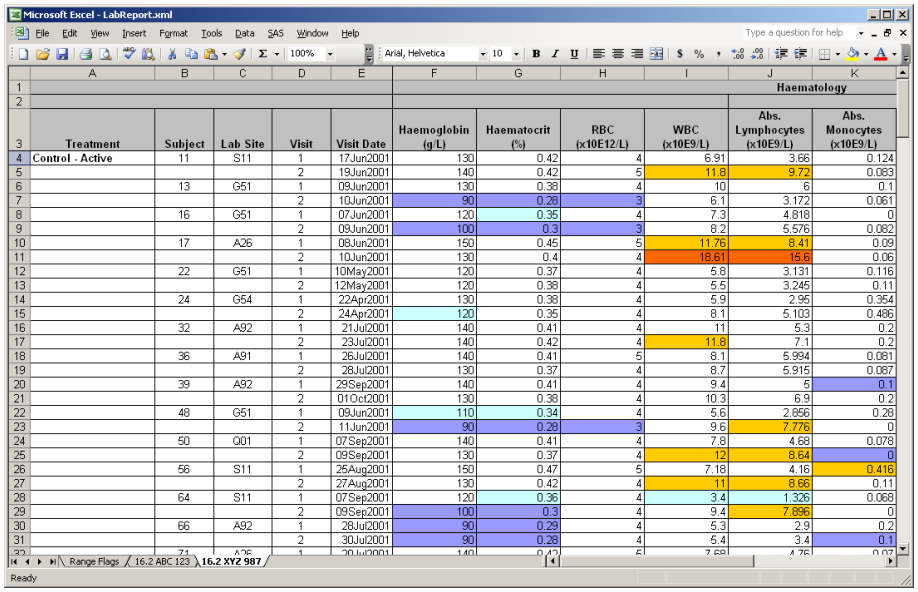
4

This is the SAS output that has been incorporated into Excel. The workbook was created using the PRINT and REPORT procedures, and contains fictional lab result data for clinical trials "ABC 123" and "XYZ 987".

This worksheet serves as a legend, defining the meaning of the traffic light colors, and was created using the PRINT procedure.

The techniques used to traffic light this data also work with the HTML, RTF and PDF destinations.

PROC REPORT Output ⟶ Excel



5

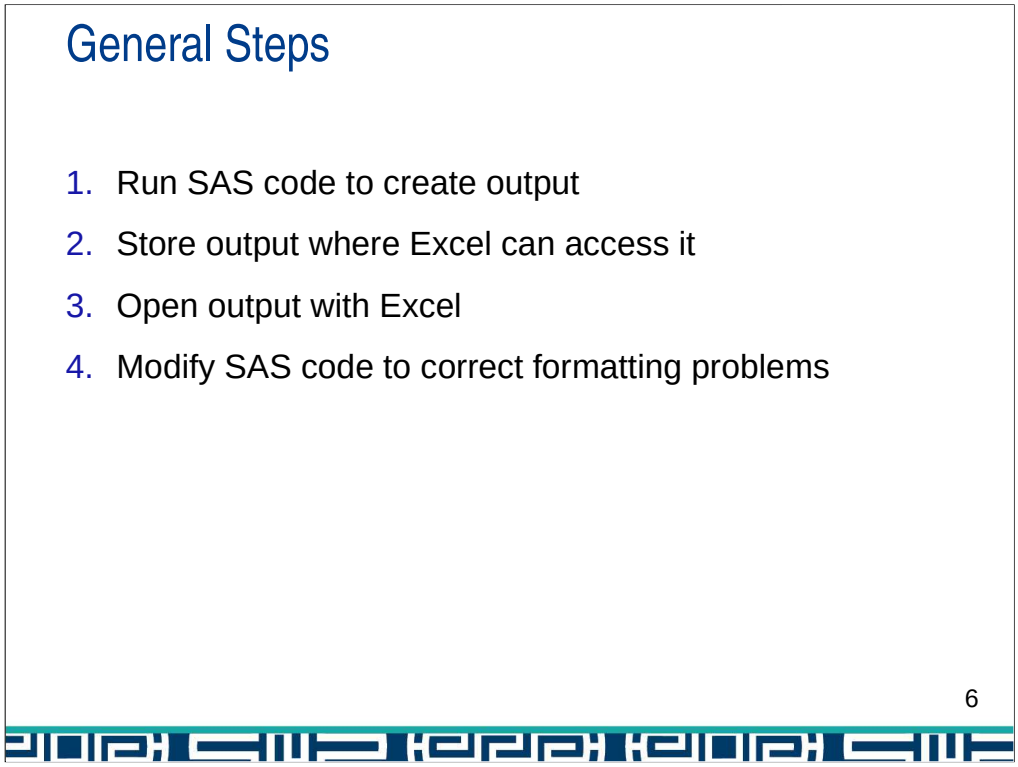
The remaining two worksheets were created using the REPORT procedure, and contain traffic-lighted lab result data.

Cells with non-white backgrounds are used to traffic light values that are outside the normal range.

Excel formats, not SAS formats, control the appearance of the date values. The date values are SAS date values that have been converted to Excel date values, without modifying the underlying SAS data.

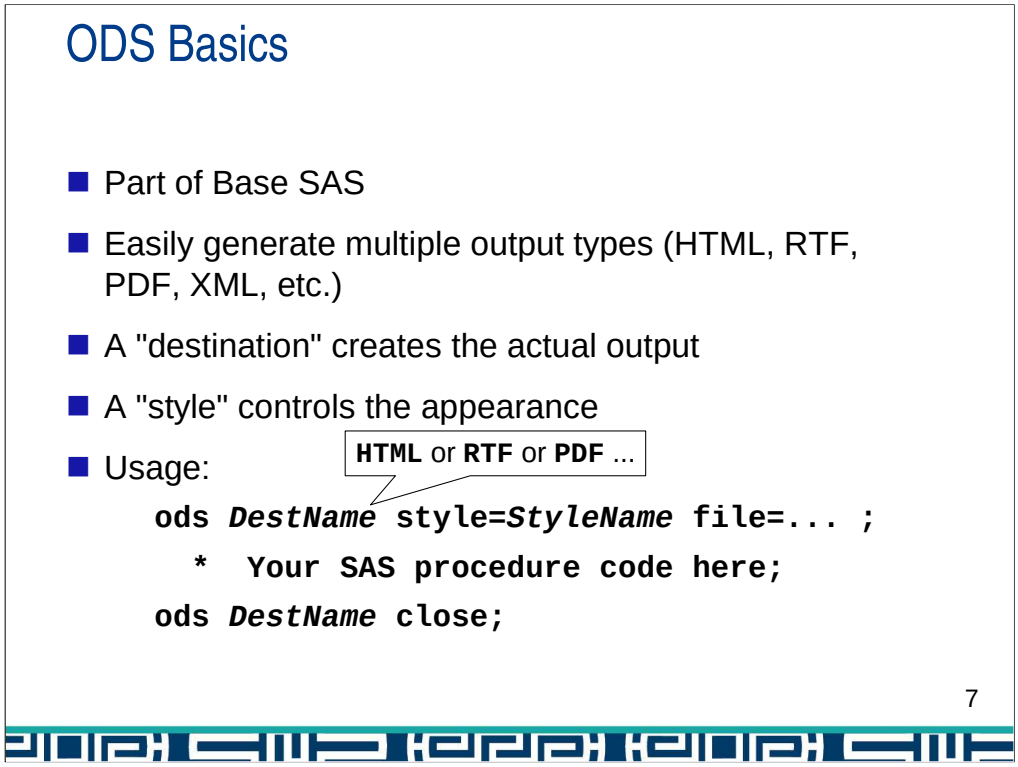
This workbook was generated completely by ODS – there was no "hand-editing" of the Excel workbook. Furthermore, SAS can generate this type of output regardless of the platform – Windows, UNIX, OpenVMS, or z/OS.

The techniques used to traffic light this data 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

ODS Basics – Output for Excel

- Excel XP can open specially made XML files as multi-sheet workbooks (graphics not supported)
- Use the ExcelXP tagset and XLsansPrinter style:

```
ods listing close;
ods tagsets.ExcelXP style=XLsansPrinter
 file=... ;
   *  PROC PRINT code here;
   *  PROC REPORT code here;
ods tagsets.ExcelXP close;
```

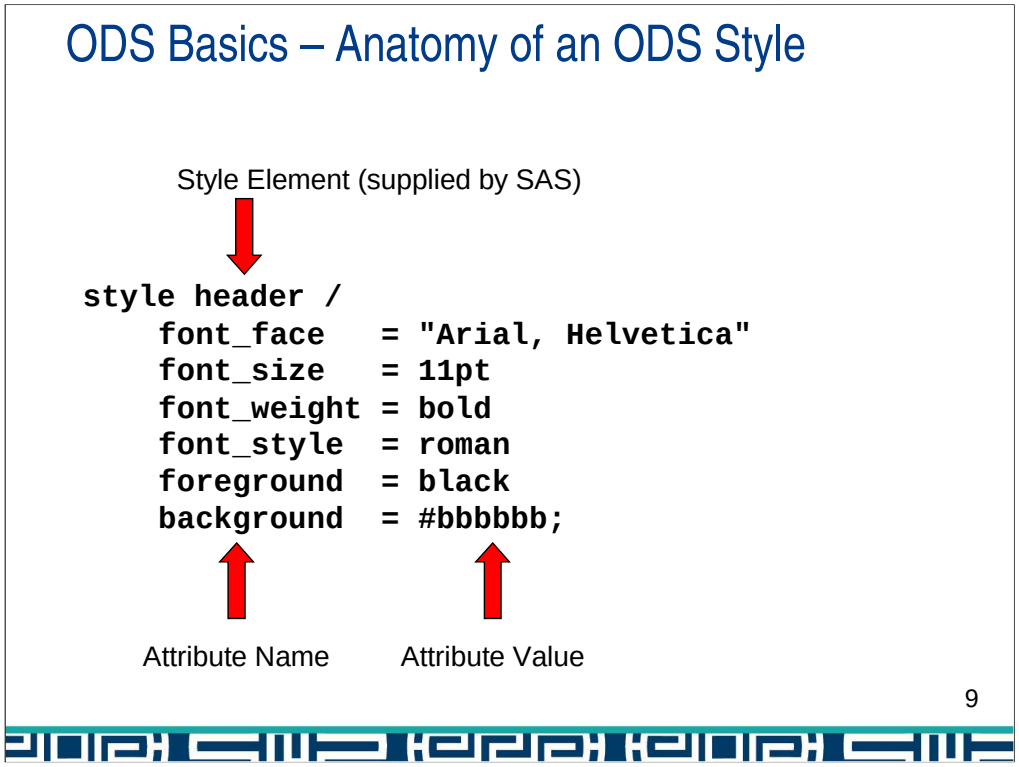
8

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:

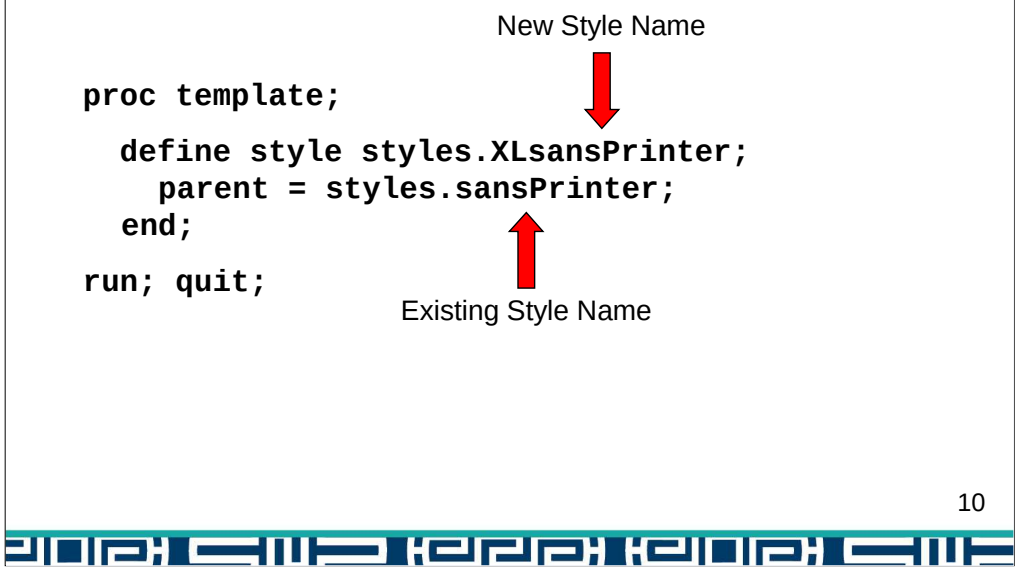
SAS 9.1:

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

ODS Basics – Anatomy of an ODS Style



10

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.

ODS Basics – Anatomy of an ODS Style

Syntax for creating style elements:

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

new-element inherits attributes from ***existing-element***

11

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

ODS Basics – Anatomy of an ODS Style

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

New Style Element

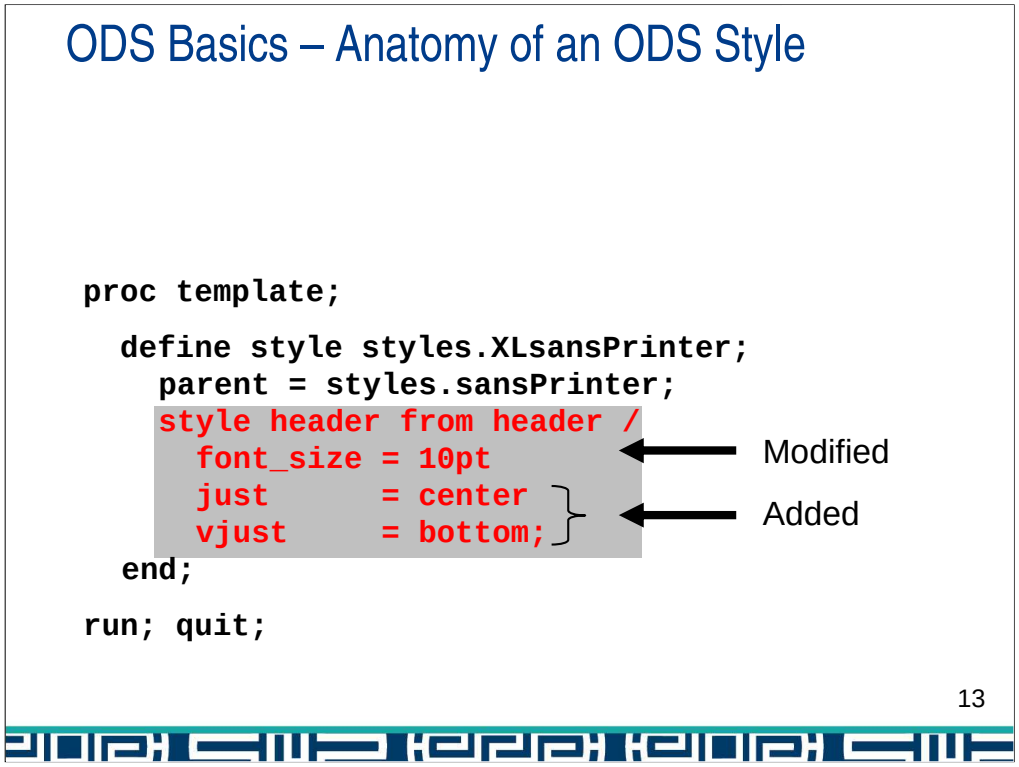
Existing (Parent) Style Element

12

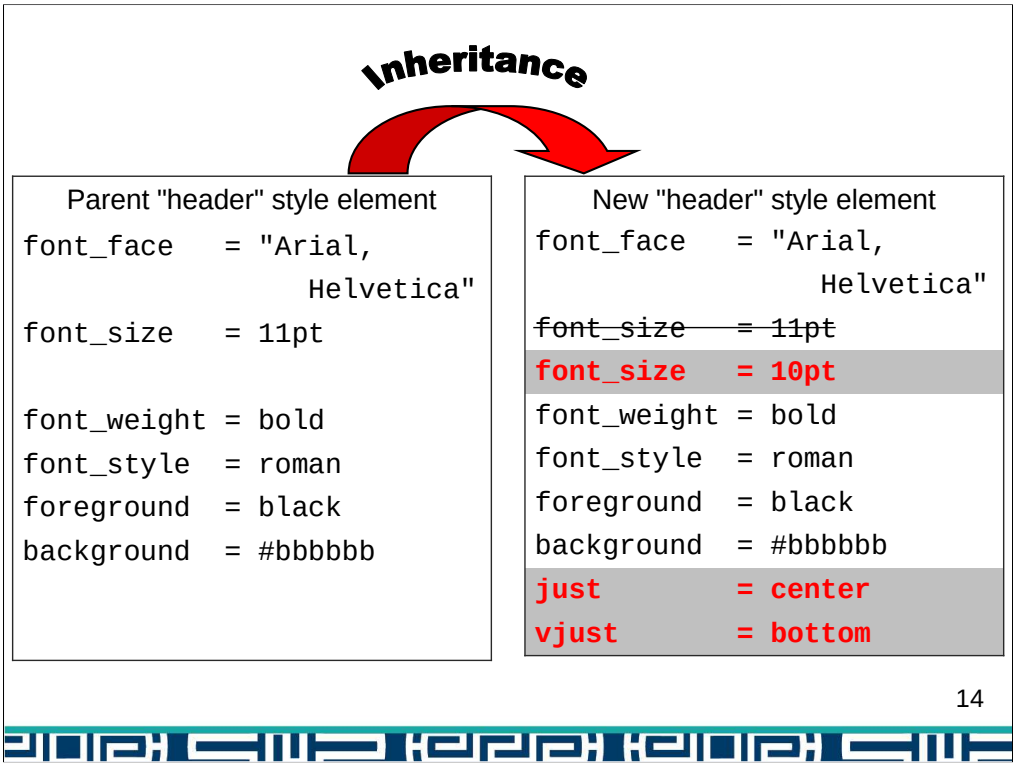
Here is an example of how you would modify the "header" style element that is supplied by SAS. The new "header" style element inherits all the style attributes of the existing (parent) "header" style element shown earlier.

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.

ODS Basics – Anatomy of an ODS Style

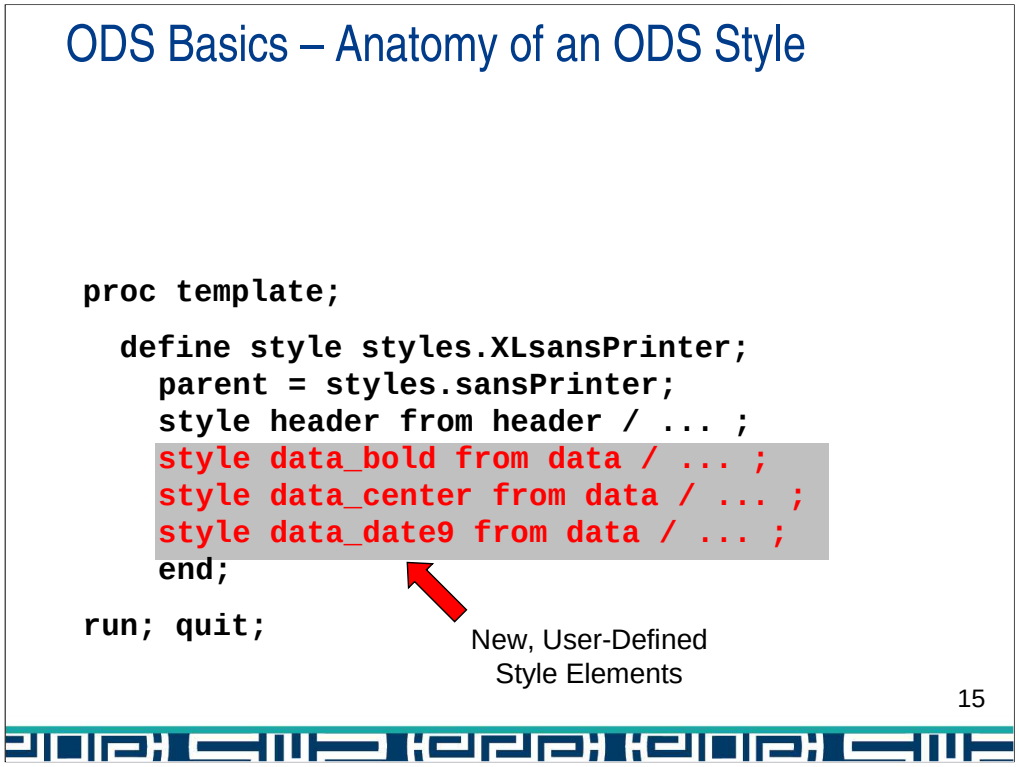


This code illustrates how you can change the value of an existing style attribute ("font_size"), and also add 2 new attributes ("just" and "vjust"). All other attributes of this style element, such as the foreground color, are inherited from the parent "header" style element, and remain unchanged.



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.



This code creates 3 new, user-defined style elements, "data_bold", "data_center" and "data_date9". Each of these style elements inherit all the style attributes of the "data" style element that is supplied by SAS.

Further information about the attributes associated with these style elements will be presented later.

Because of their unique names, ODS does not use these style elements until they are specified by you via a style override.

We will now move on to the topic of style overrides.

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

ODS Basics – Style Overrides

- Supported by PRINT, TABULATE and REPORT
- Change any ODS style attribute via STYLE=
- Example:

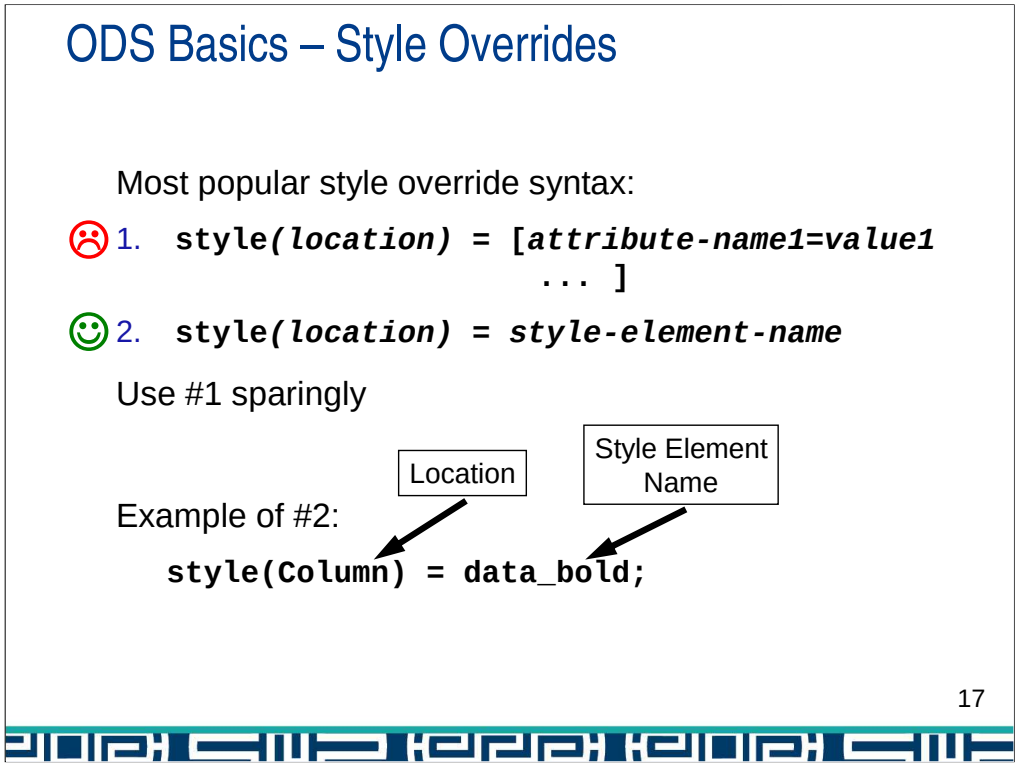
```
style(Column) = [font_style=italic
                 background=blue]
```

- Refer to the ODS documentation for a list of supported attributes
- Refer to PRINT, TABULATE and REPORT doc for sample usage

16

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 PRINT, as shown here.



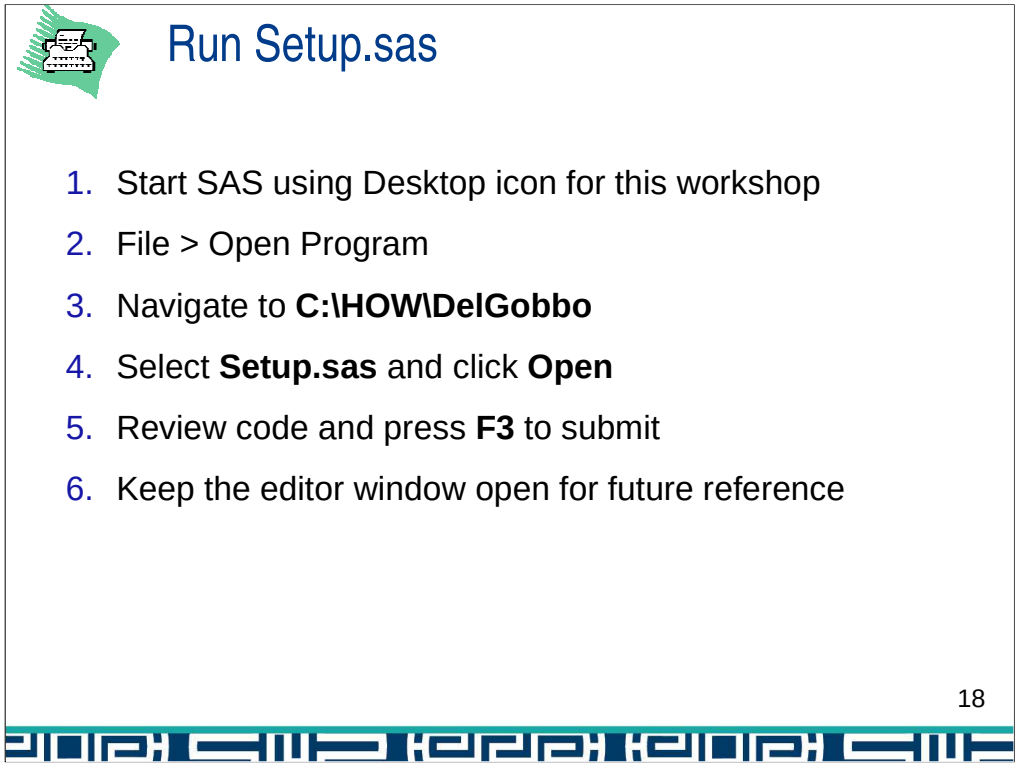
Here is the syntax for using style overrides.

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 avoided when possible. We use this method only to traffic light cells in the PRINT and REPORT procedure output.

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 XLsansPrinter style defined in the "Setup.sas" file contains a number of different style elements. Later, we associate these style elements with discrete parts of the SAS output ("locations").

Locations will be discussed later.



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" and "notecontent" style elements supplied by SAS are modified, and new, user-defined style elements "data_bold", "data_center" and "data_date9" are created.

Finally, the SAS table "Legend" and SAS format "FlagFmt", which is used to perform traffic lighting, are created.

Sample SAS Data – Lab Results

Variable Name	Typical Values
Protocol	ABC 123 or XYZ 987
Treat	Control – Active, Test – High Dose
Patient	1 – 140
LabSite	S11, G51, A26
Visit	1 or 2
VisitDate	2001-06-17, 2001-06-19
Haemoglobin_Result	60 – 160
Haemoglobin_Flag	0, 1, 2, 3, or 4
Haematocrit_Result	2.0 – 6.0
Haematocrit_Flag	0, 1, 2, 3, or 4
. . .	. . .

19

The "LabResults" SAS table contains the lab result data that is displayed using PROC REPORT.

The variable "Protocol" represents the name of the study plan for the clinical trial, and contains the value "ABC 123" or "XYZ 987". This variable is used in a BY statement in our PROC REPORT code, and the BY group values are used in the resulting worksheet names.

"VisitDate" is a SAS date value with the SAS YYMMDD10. format applied.

Variables whose names end in "_Result" contain the lab result values to traffic light. The table above lists 4 of the 10 such variables displayed in our Excel workbook. The background color of the cell is determined by the value of the corresponding "_Flag" variable. For example, the value of the variable "Haemoglobin_Flag" determines the background color for "Haemoglobin_Result".

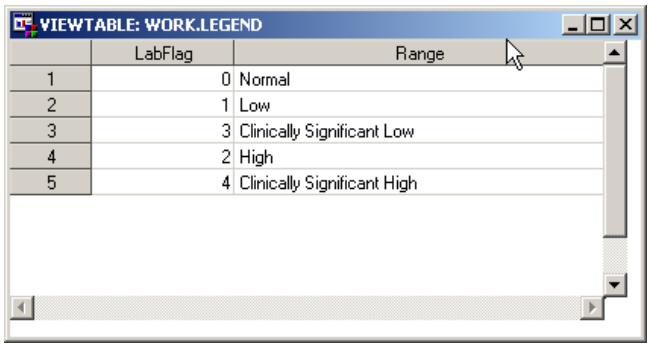
Traffic Lighting Criteria

Flag Value	Flag Meaning	Background Color
0	Normal	white
1	Low	#CCFFFF
3	Clinically Significant Low	#9999FF
2	High	#FFCC00
4	Clinically Significant High	#FF6600

20

This table shows the background colors corresponding to the values of the "_Flag" variables.

Sample SAS Data – Range Flags

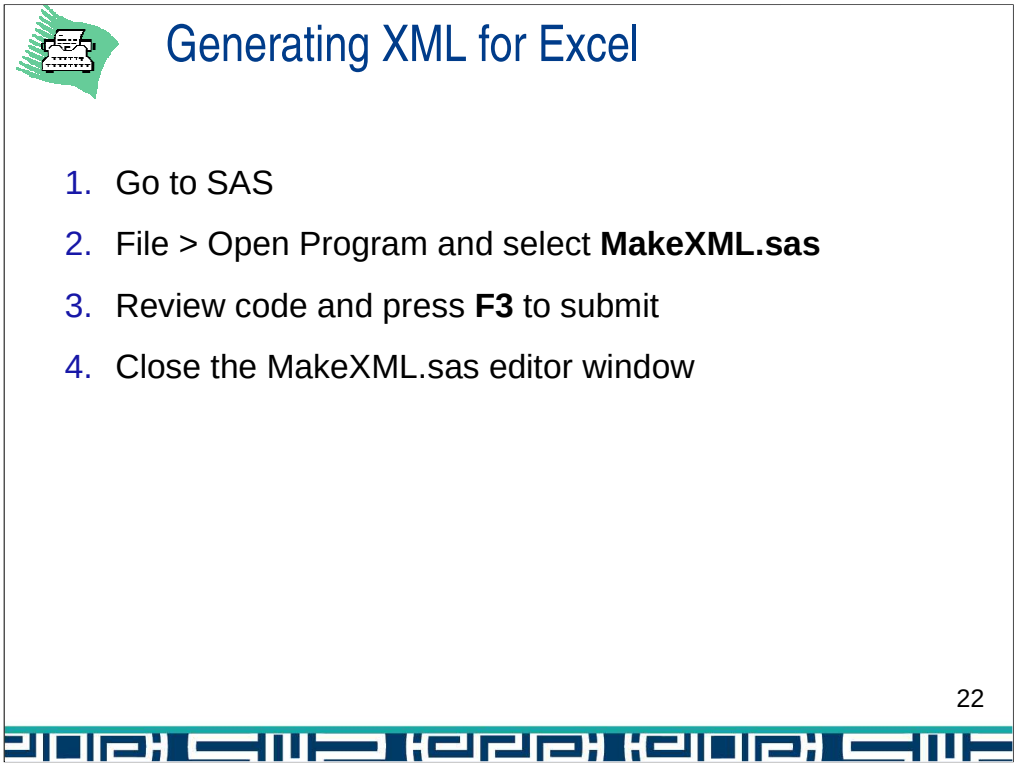


	LabFlag	Range
1	0	Normal
2	1	Low
3	3	Clinically Significant Low
4	2	High
5	4	Clinically Significant High

21

The "Legend" SAS table contains data that is used as a legend, defining the meaning of the traffic light colors. This data is displayed using the PRINT procedure.

The data values "LabFlag" will be traffic lighted using the criteria shown on the previous slide.



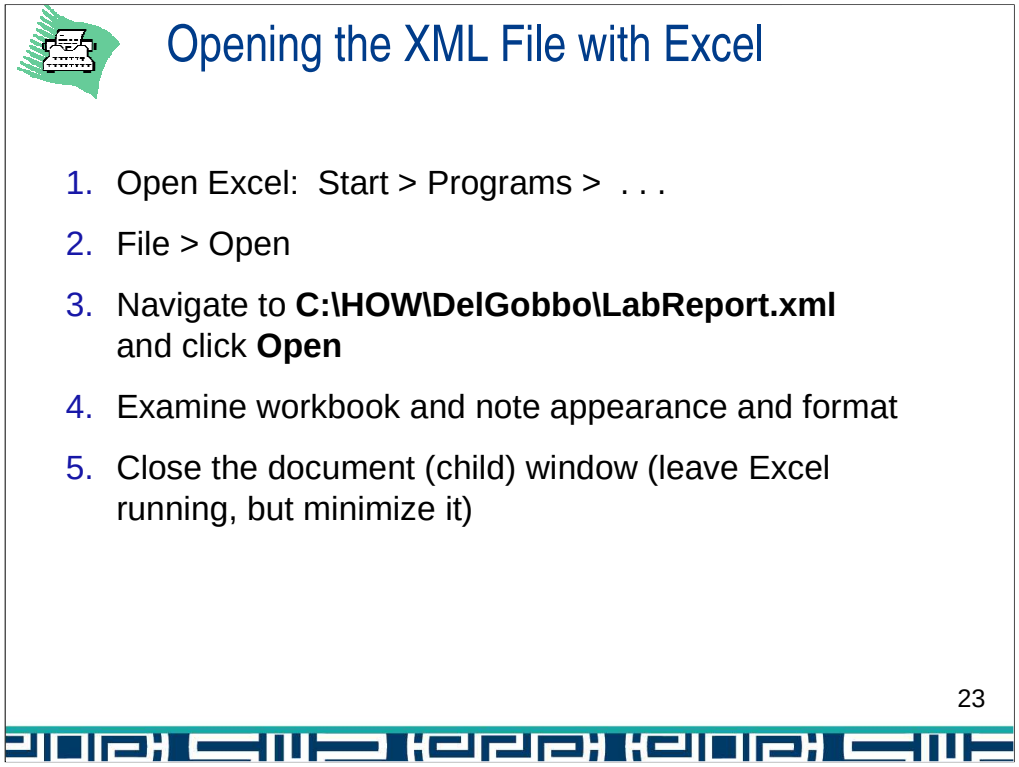
This is our first attempt to make an XML file that can be opened with Excel.

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 "LabReport.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 PRINT procedure creates the range flags worksheet and the REPORT procedure is run with a BY statement to create worksheets 2 and 3, one worksheet for each distinct value of the variable "Protocol".

The first COMPUTE block is used to traffic light the lab result data, and the second one inserts white space between the different treatment values.



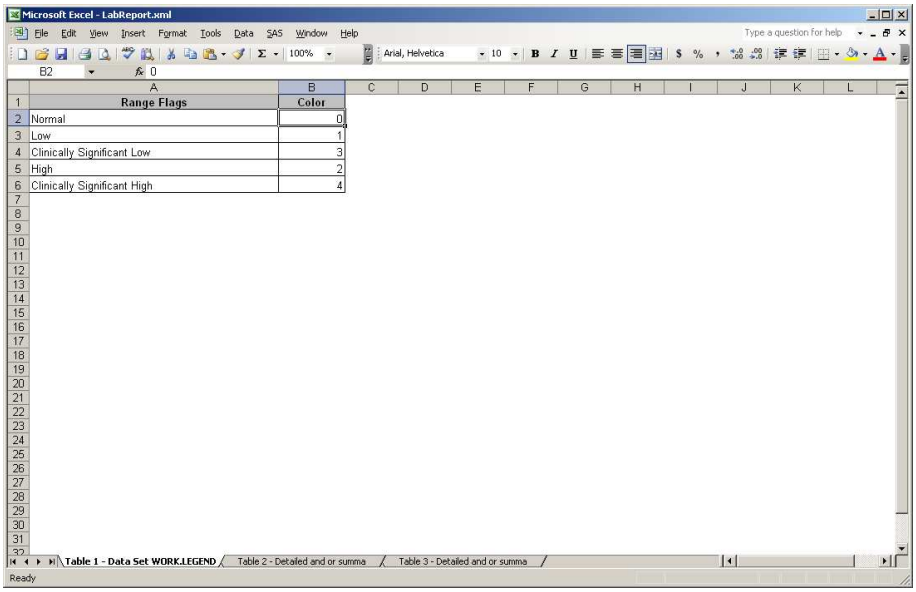
The XML file created by ODS can now be opened with Microsoft Excel.

When Excel reads the XML file, it renders it as a multi-sheet workbook. Thus you can use all the editing and formatting features of Excel to modify the output and, if desired, save it as a native Excel workbook in binary format.

To save a copy of the file in Excel binary format using Excel 2002 or Excel 2003, 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

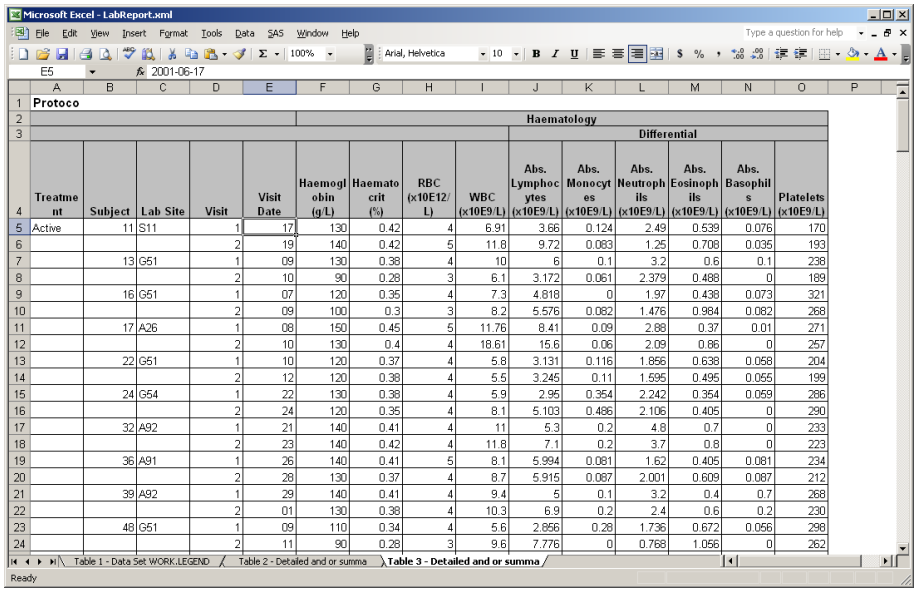


24

Problems with the output:

1. None of the values in any of the worksheets are traffic-lighted.
2. Data values in the "Treatment" column are not bold text.
3. Values in the "Subject", "Lab Site" and "Visit" columns are not centered.
4. The date values are displayed incorrectly.
5. Standard BY group text ("Protocol Identifier=ABC 123" or "Protocol Identifier=XYZ 987") precedes the REPORT procedure output and is obscured.
6. Unattractive, default worksheet names are used.
7. Some of the columns created by REPORT procedure are too narrow.
8. If you were to scroll downward or to the right in this worksheet, you will "lose" the column and row headings because by default, they are not "frozen".

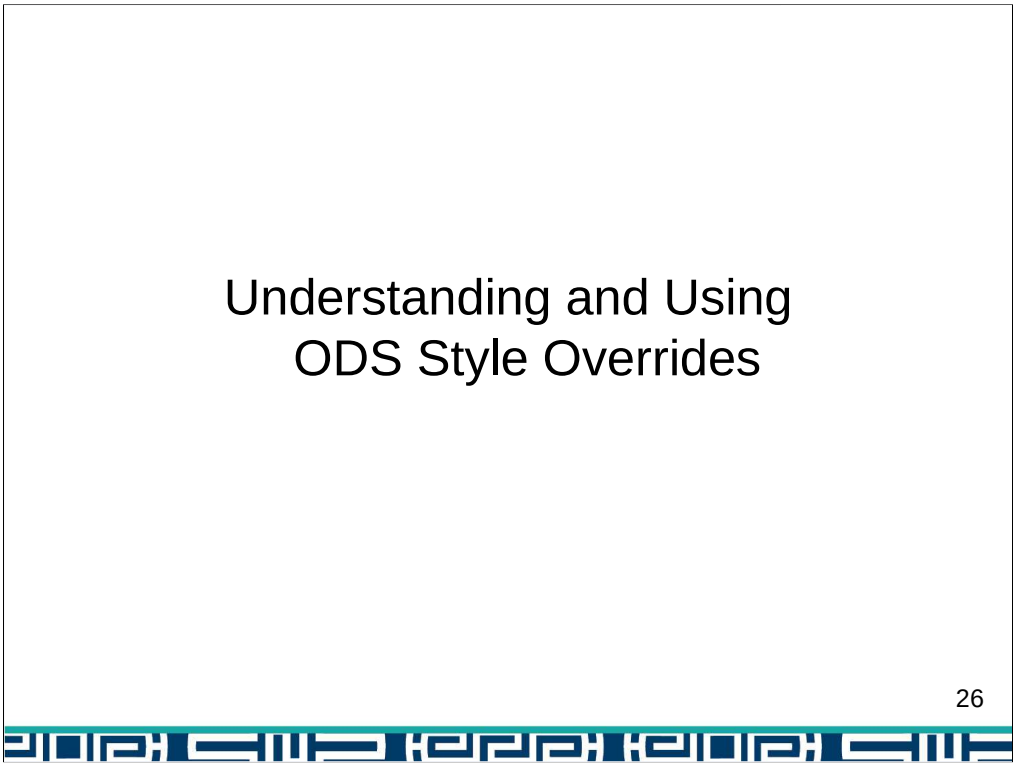
XML Output Viewed Using Excel



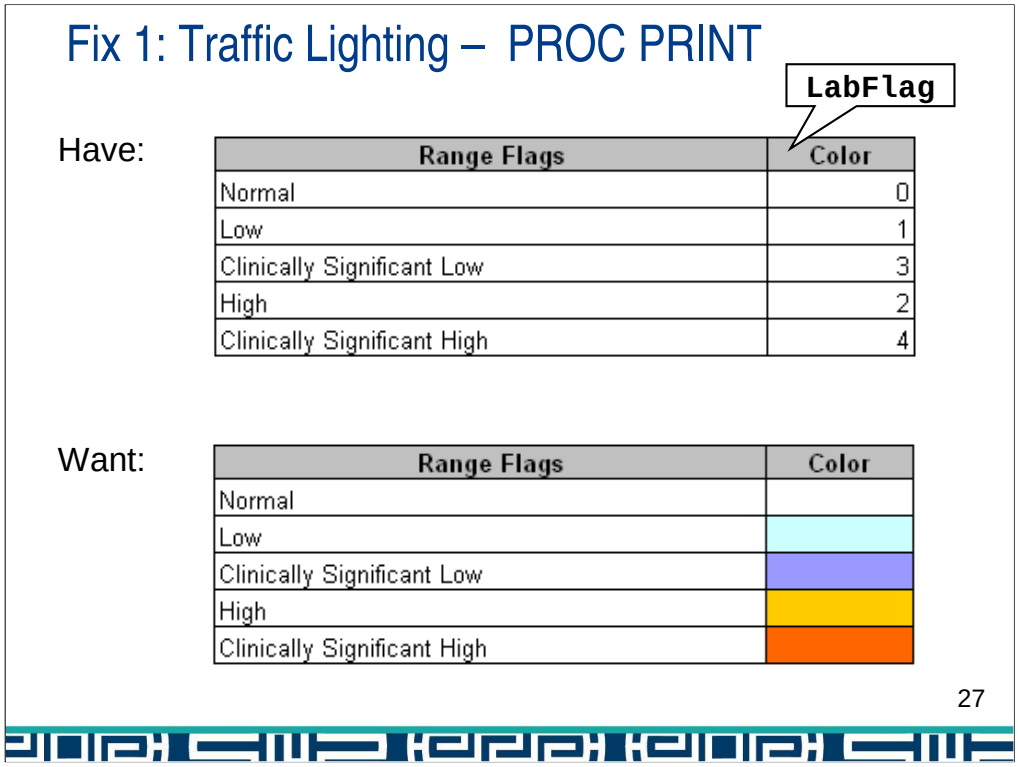
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Protoco															
2						Haematology										
3										Differential						
4	Treatment	Subject	Lab Site	Visit	Visit Date	Haemoglobin (g/L)	Haematocrit (%)	RBC (x10E12/L)	WBC (x10E9/L)	Abs. Lymphocytes (x10E9/L)	Abs. Monocytes (x10E9/L)	Abs. Neutrophils (x10E9/L)	Abs. Eosinophils (x10E9/L)	Abs. Basophils (x10E9/L)	Platelets (x10E9/L)	
5	Active	11	S11	1	17	130	0.42	4	6.91	3.66	0.124	2.49	0.539	0.076	170	
6				2	19	140	0.42	5	11.8	9.72	0.083	1.25	0.708	0.035	193	
7		13	G51	1	09	130	0.38	4	10	6	0.1	3.2	0.6	0.1	238	
8				2	10	90	0.28	3	6.1	3.172	0.061	2.379	0.488	0	189	
9		16	G51	1	07	120	0.35	4	7.3	4.818	0	1.97	0.438	0.073	321	
10				2	09	100	0.3	3	8.2	5.576	0.082	1.476	0.984	0.082	268	
11		17	A26	1	08	150	0.45	5	11.76	8.41	0.09	2.88	0.37	0.01	271	
12				2	10	130	0.4	4	18.61	15.6	0.06	2.09	0.86	0	257	
13		22	G51	1	10	120	0.37	4	5.8	3.131	0.116	1.856	0.638	0.058	204	
14				2	12	120	0.38	4	5.5	3.245	0.11	1.595	0.495	0.055	199	
15		24	G54	1	22	130	0.38	4	5.9	2.95	0.354	2.242	0.354	0.059	286	
16				2	24	120	0.35	4	8.1	5.103	0.486	2.106	0.405	0	290	
17		32	A92	1	21	140	0.41	4	11	5.3	0.2	4.8	0.7	0	233	
18				2	23	140	0.42	4	11.8	7.1	0.2	3.7	0.8	0	223	
19		36	A91	1	26	140	0.41	5	8.1	5.994	0.081	1.62	0.405	0.081	234	
20				2	28	130	0.37	4	8.7	5.915	0.087	2.001	0.609	0.087	212	
21		39	A92	1	29	140	0.41	4	9.4	5	0.1	3.2	0.4	0.7	268	
22				2	01	130	0.38	4	10.3	6.9	0.2	2.4	0.6	0.2	230	
23		48	G51	1	09	110	0.34	4	5.6	2.856	0.28	1.736	0.672	0.056	298	
24				2	11	90	0.28	3	9.6	7.776	0	0.768	1.056	0	262	

25

One of the lab results worksheets.



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



In order to achieve the desired result, the foreground and background colors of the "LabFlag" column need to be the same. For example, we want white text ("0") on a white background.

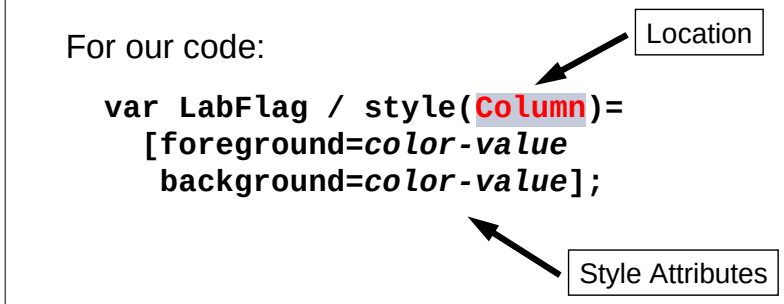
This can be accomplished using an ODS style override.

Fix 1: Traffic Lighting – PROC PRINT

General style override syntax:

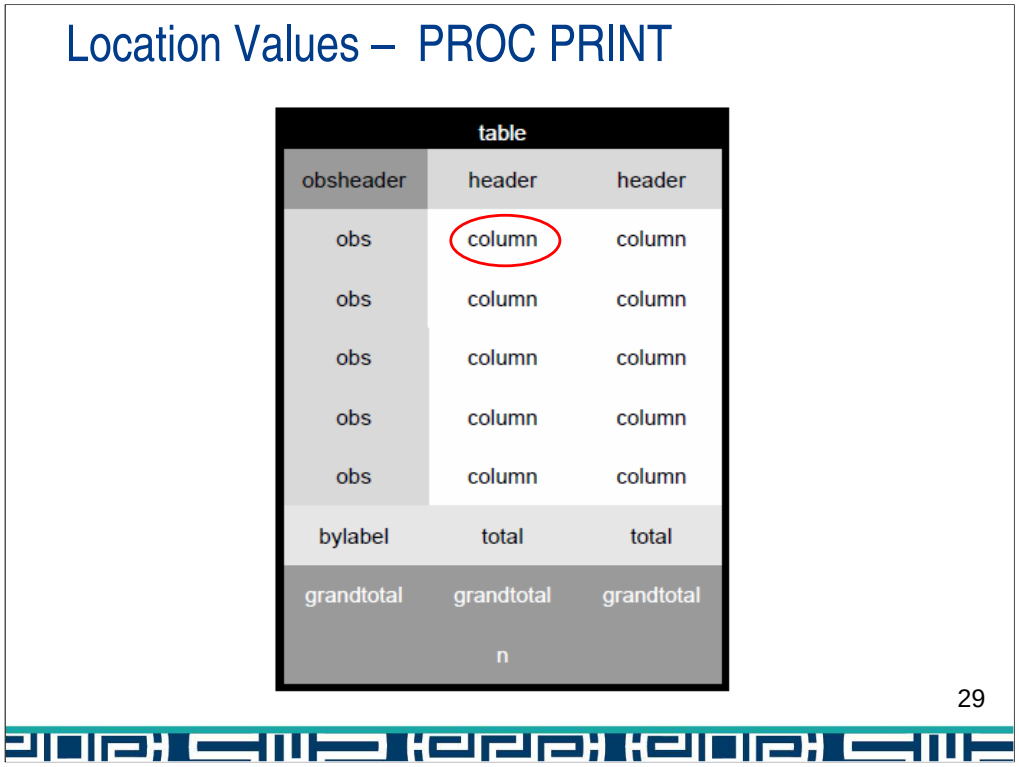
```
style(location)=style-specification
```

For our code:



28

The syntax for a style override is shown above. Note that we are using method 1 from slide 17 (individual style attributes defined 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 PRINT output.

To change the appearance of the data cells in the "LabFlag" column, we apply a style override to the "Column" location.

Reference:

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

Fix 1: Traffic Lighting – PROC PRINT

```
var LabFlag / style(Column)=
  [foreground=FlagFmt.
   background=FlagFmt.];

proc format;
  value FlagFmt 0 = 'white'
                1 = '#CCFFFF'
                3 = '#9999FF'
                2 = '#FFCC00'
                4 = '#FF6600';
```

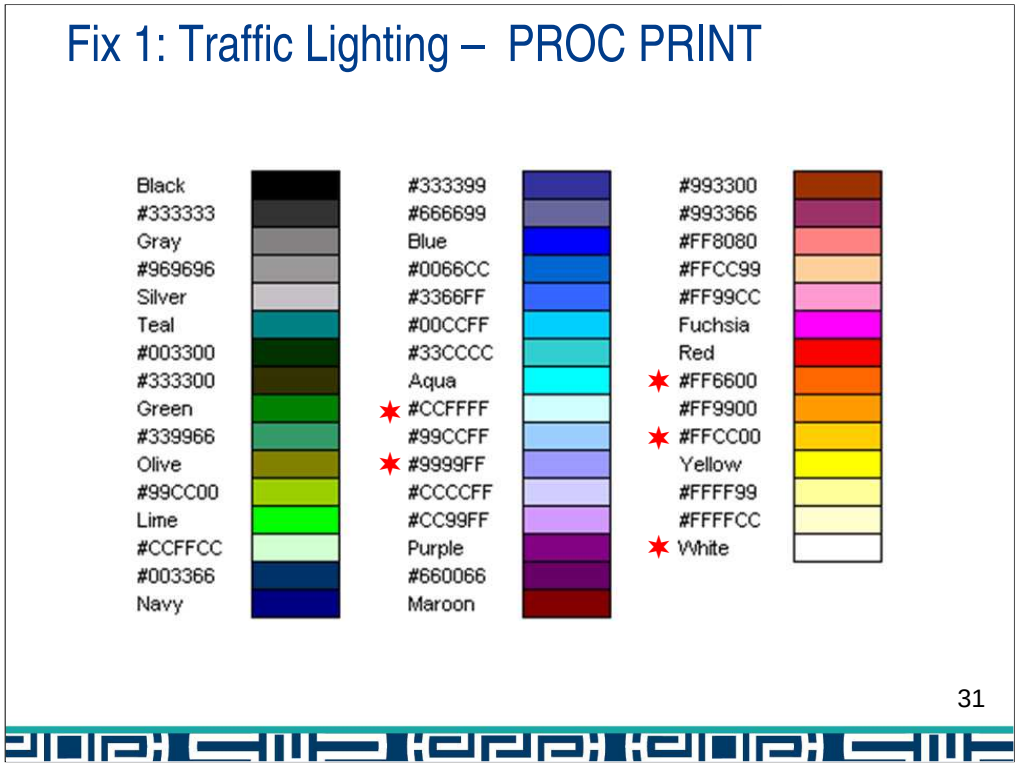
SAS Format Name

30

We cannot simply specify a color value for the "foreground" and "background" attributes, because *all* of the cells in the "LabFlag" column would have the same foreground and background color. Instead, we need the colors to change based on the value in the cell.

Specifying a SAS format instead of a color value causes the current value of the data in the cell to be evaluated against the format to determine the attribute's value. For example, cells containing the value "1" have a light turquoise (#CCFFFF) foreground and background.

The "FlagFmt" format was constructed based on the criteria shown on slide 20.



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 by the "FlagFmt" format 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 does not have this restricted color palette.

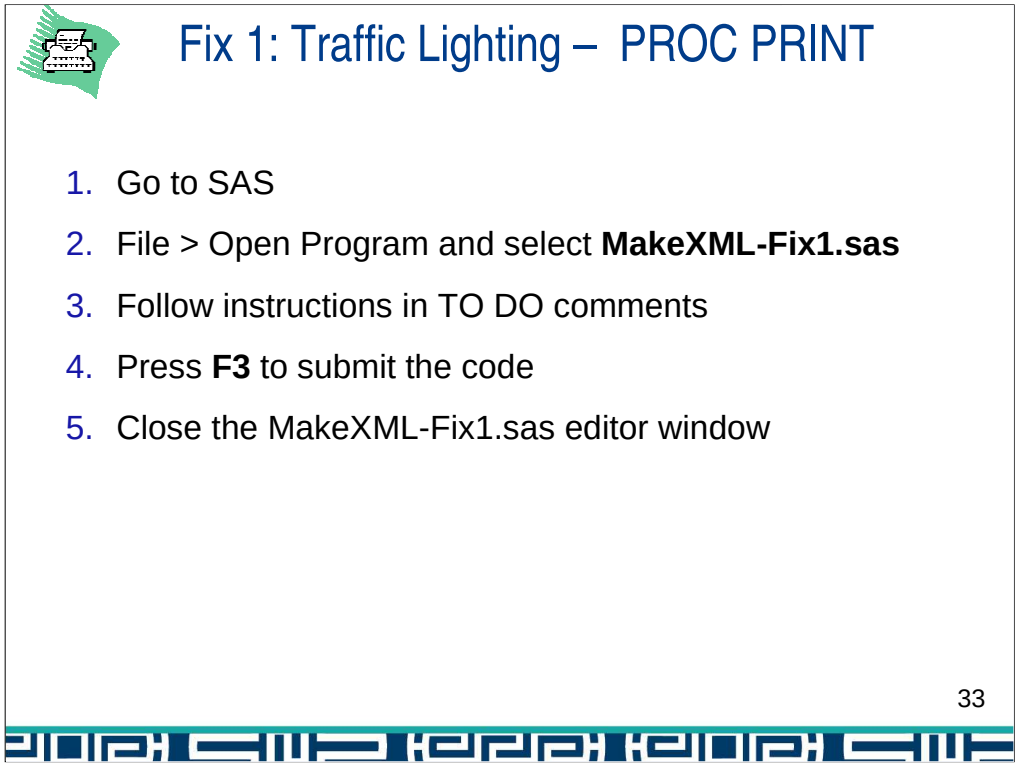
Fix 1: Traffic Lighting – PROC PRINT

```
proc print data=Legend noobs label;
  var Range;
  var LabFlag / style(Column)=
    [foreground=FlagFmt.
     background=FlagFmt.];
run; quit;
```

32

Here is the code needed to traffic light the PROC PRINT output.

This technique also works with the HTML, RTF and PDF destinations.



TO DO:

Line 24: Specify the SAS format name for the FOREGROUND and BACKGROUND style attributes.

Solution:

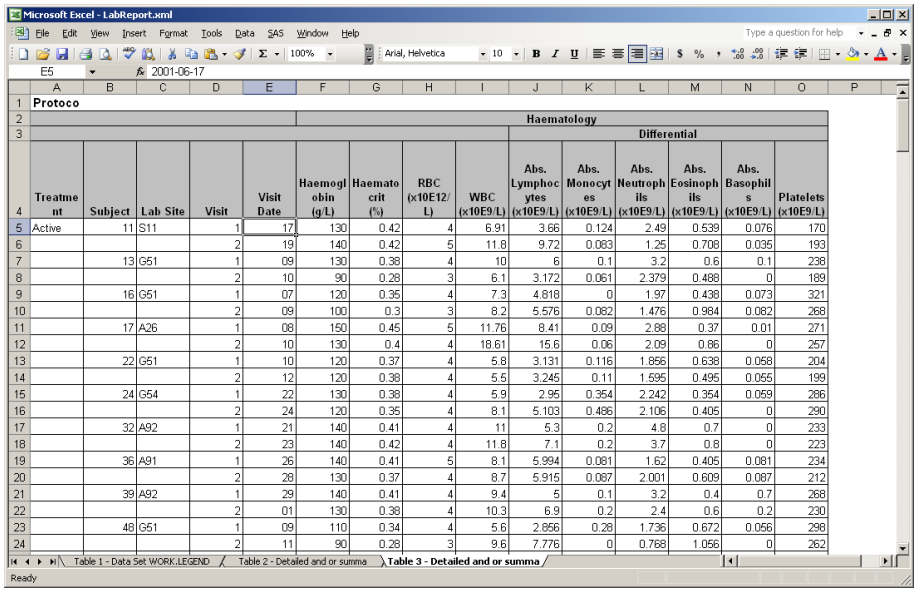
```
var labflag / style(Column)=[foreground=FlagFmt.
                             background=FlagFmt.];
```

Fix 1: Traffic Lighting – PROC PRINT

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

34

Fix 2: Traffic Lighting – PROC REPORT

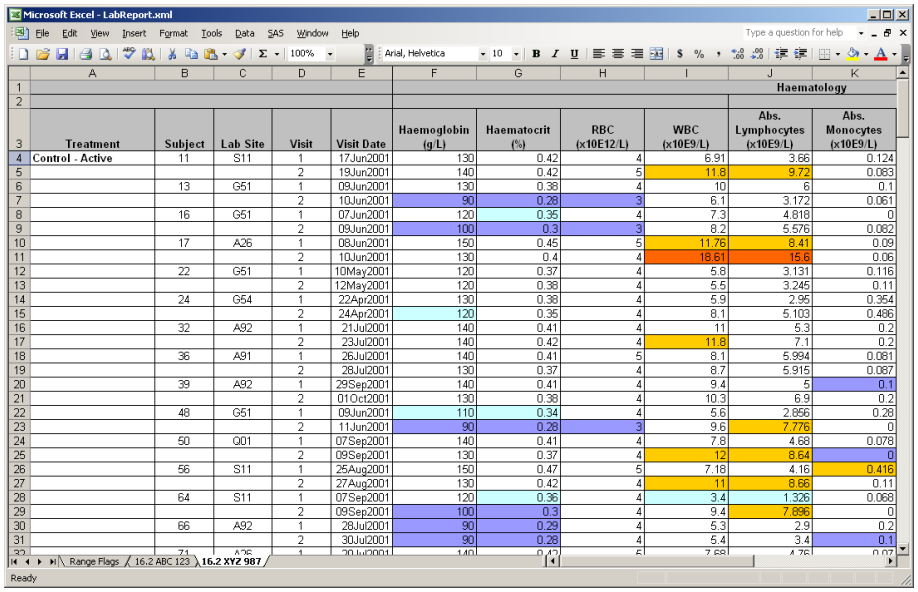


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Protoco															
2						Haematology										
3										Differential						
4	Treatment	Subject	Lab Site	Visit	Visit Date	Haemoglobin (g/L)	Haematocrit (%)	RBC (x10E12/L)	WBC (x10E9/L)	Abs. Lymphocytes (x10E9/L)	Abs. Monocytes (x10E9/L)	Abs. Neutrophils (x10E9/L)	Abs. Eosinophils (x10E9/L)	Abs. Basophils (x10E9/L)	Platelets (x10E9/L)	
5	Active	11	S11	1	17	130	0.42	4	6.91	3.66	0.124	2.49	0.539	0.076	170	
6				2	19	140	0.42	5	11.8	9.72	0.083	1.25	0.708	0.035	193	
7		13	G51	1	09	130	0.38	4	10	6	0.1	3.2	0.6	0.1	238	
8				2	10	90	0.28	3	6.1	3.172	0.061	2.379	0.488	0	189	
9		16	G51	1	07	120	0.35	4	7.3	4.818	0	1.97	0.438	0.073	321	
10				2	09	100	0.3	3	8.2	5.576	0.082	1.476	0.984	0.082	268	
11		17	A26	1	08	150	0.45	5	11.76	8.41	0.09	2.88	0.37	0.01	271	
12				2	10	130	0.4	4	18.61	15.6	0.06	2.09	0.86	0	257	
13		22	G51	1	10	120	0.37	4	5.8	3.131	0.116	1.856	0.638	0.058	204	
14				2	12	120	0.38	4	5.5	3.245	0.11	1.595	0.495	0.055	199	
15		24	G54	1	22	130	0.38	4	5.9	2.95	0.354	2.242	0.354	0.059	286	
16				2	24	120	0.35	4	8.1	5.103	0.486	2.106	0.405	0	290	
17		32	A92	1	21	140	0.41	4	11	5.3	0.2	4.8	0.7	0	233	
18				2	23	140	0.42	4	11.8	7.1	0.2	3.7	0.8	0	223	
19		36	A91	1	26	140	0.41	5	8.1	5.994	0.081	1.62	0.405	0.081	234	
20				2	28	130	0.37	4	8.7	5.915	0.087	2.001	0.609	0.087	212	
21		39	A92	1	29	140	0.41	4	9.4	5	0.1	3.2	0.4	0.7	268	
22				2	01	130	0.38	4	10.3	6.9	0.2	2.4	0.6	0.2	230	
23		48	G51	1	09	110	0.34	4	5.6	2.856	0.28	1.736	0.672	0.056	298	
24				2	11	90	0.28	3	9.6	7.776	0	0.768	1.056	0	262	

35

This is what we have.

Fix 2: Traffic Lighting – PROC REPORT



	A	B	C	D	E	F	G	H	I	J	K
1										Haematology	
2											
3	Treatment	Subject	Lab Site	Visit	Visit Date	Haemoglobin (g/L)	Haematocrit (%)	RBC (x10E12/L)	WBC (x10E9/L)	Abs. Lymphocytes (x10E9/L)	Abs. Monocytes (x10E9/L)
4	Control - Active	11	S11	1	17Jun2001	130	0.42	4	6.91	3.66	0.124
5				2	19Jun2001	140	0.42	5	11.8	9.72	0.083
6		13	G51	1	09Jun2001	130	0.38	4	10	6	0.1
7				2	10Jun2001	90	0.28	3	6.1	3.172	0.061
8		16	G51	1	07Jun2001	120	0.35	4	7.3	4.818	0
9				2	09Jun2001	100	0.3	3	8.2	5.576	0.082
10		17	A26	1	08Jun2001	150	0.45	5	11.76	8.41	0.09
11				2	10Jun2001	130	0.4	4	18.61	15.6	0.06
12		22	G51	1	10May2001	120	0.37	4	5.8	3.131	0.116
13				2	12May2001	120	0.38	4	5.5	3.245	0.11
14		24	G54	1	22Apr2001	130	0.36	4	5.9	2.95	0.354
15				2	24Apr2001	120	0.35	4	8.1	5.103	0.486
16		32	A92	1	21Jul2001	140	0.41	4	11	5.3	0.2
17				2	23Jul2001	140	0.42	4	11.8	7.1	0.2
18		36	A91	1	26Jul2001	140	0.41	5	8.1	5.994	0.081
19				2	28Jul2001	130	0.37	4	8.7	5.916	0.087
20		39	A92	1	29Sep2001	140	0.41	4	9.4	5	0.1
21				2	01Oct2001	130	0.38	4	10.3	6.9	0.2
22		48	G51	1	09Jun2001	110	0.34	4	5.6	2.856	0.28
23				2	11Jun2001	90	0.28	3	9.6	7.776	0
24		50	Q01	1	07Sep2001	140	0.41	4	7.8	4.68	0.078
25				2	09Sep2001	130	0.37	4	12	8.64	0
26		56	S11	1	25Aug2001	150	0.47	5	7.18	4.16	0.416
27				2	27Aug2001	130	0.42	4	11	8.66	0.11
28		64	S11	1	07Sep2001	120	0.36	4	3.4	1.326	0.068
29				2	09Sep2001	100	0.3	4	9.4	7.896	0
30		66	A92	1	28Jul2001	90	0.29	4	5.3	2.9	0.2
31				2	30Jul2001	90	0.28	4	5.4	3.4	0.1

This is what we want. The lab results are traffic-lighted based on the value of the corresponding _Flag variable (not shown in the worksheets).

Fix 2: Traffic Lighting – PROC REPORT

Use CALL DEFINE:

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

Example:

```
call define('Haemoglobin_Result',
            'style',
            'style=[background=color-value]');
```

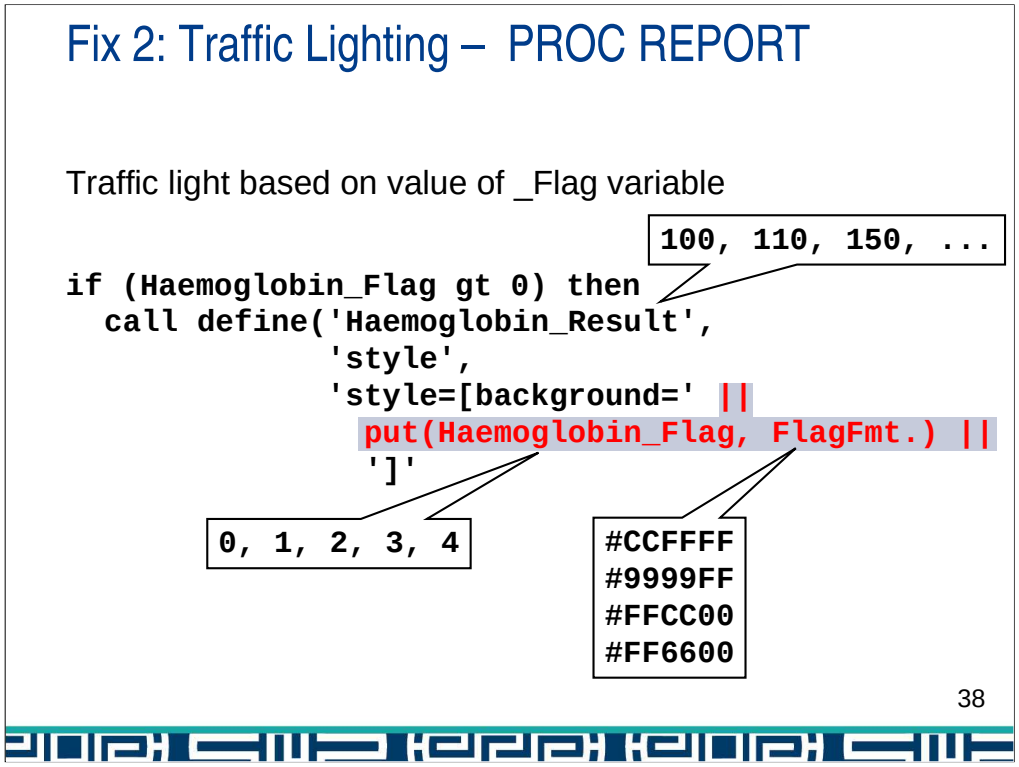
37

Traffic lighting the lab result values using the REPORT procedure requires a slightly different technique. This traffic lighting uses a CALL DEFINE statement within a COMPUTE block. The CALL DEFINE statement assigns the "background" style attribute to the cell containing the lab result value.

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



Recall that each lab result value has its own flag variable that is used to determine whether the result is in or out of range. We need to examine the value of the pertinent flag variable, and use that value to determine the color value. The "FlagFmt" format is used to determine the color value based on the flag value. For example, here is the code to traffic light "Haemoglobin_Result" cells.

This technique also works with the HTML, RTF and PDF destinations.

Fix 2: Traffic Lighting – PROC REPORT

```
'style=[background=' || put(Haemoglobin_Flag, FlagFmt.) || ']'
```

_Flag Variable Value	Value of Style Attribute
0	Code not executed
1	style=[background=#CCFFFF]
3	style=[background=#9999FF]
2	style=[background=#FFCC00]
4	style=[background=#FF6600]

39

This table shows the various style attribute values generated, based on the value of the "_Flag" variable.

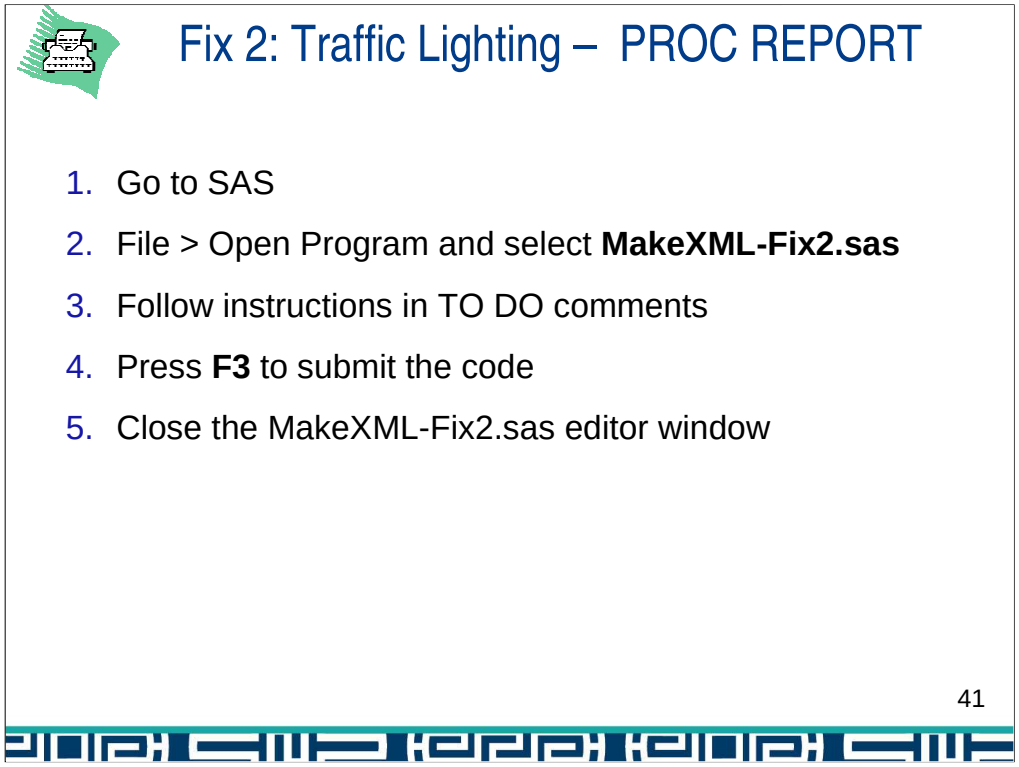
Fix 2: Traffic Lighting – PROC REPORT

Use arrays instead of repeating code:

```
array name(10) $31 ('Haemoglobin_Result'
                    'Haematocrit_Result' ...);
array flag(10) Haemoglobin_Flag
               Haematocrit_Flag ...;

do i = 1 to dim(name);
  if (flag(i) gt 0) then
    call define(name(i),
                'style',
                'style=[background' ||
                  put(flag(i), FlagFmt.) || ']');
end;
```

40

You could repeat the previous code for each of the remaining result/flag combinations, or you can make use of arrays. Using arrays provides more compact code.



TO DO:

Line 117: Specify the SAS format name for the BACKGROUND style attribute.

Solution:

```
'style=[background=' || put(flag(i), FlagFmt.) || ']'
```

Fix 2: Traffic Lighting – PROC REPORT

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

Fix 3: Bold, Center, SAS Dates → Excel Dates

User-defined style element in XLsansPrinter style:

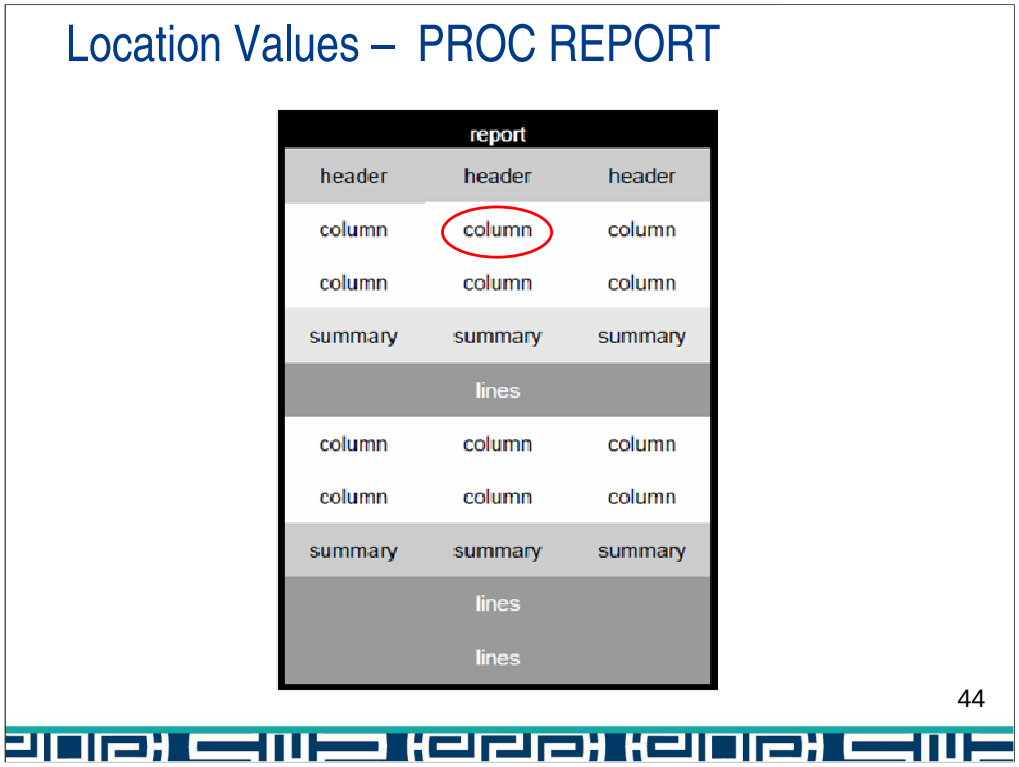
```
style data_bold from data /
  font_weight = bold;
```

43

As mentioned earlier, ODS styles control the appearance of your output. Here you see a subset of the code used to create the XLsansPrinter style. Refer to the "Setup.sas" file for a complete listing of the code.

The "data_bold" style element has all the attributes of the "data" style element that is supplied by SAS, the difference being that text is bold.

A style override is used to apply the "data_bold" style element to the "Treat" column.

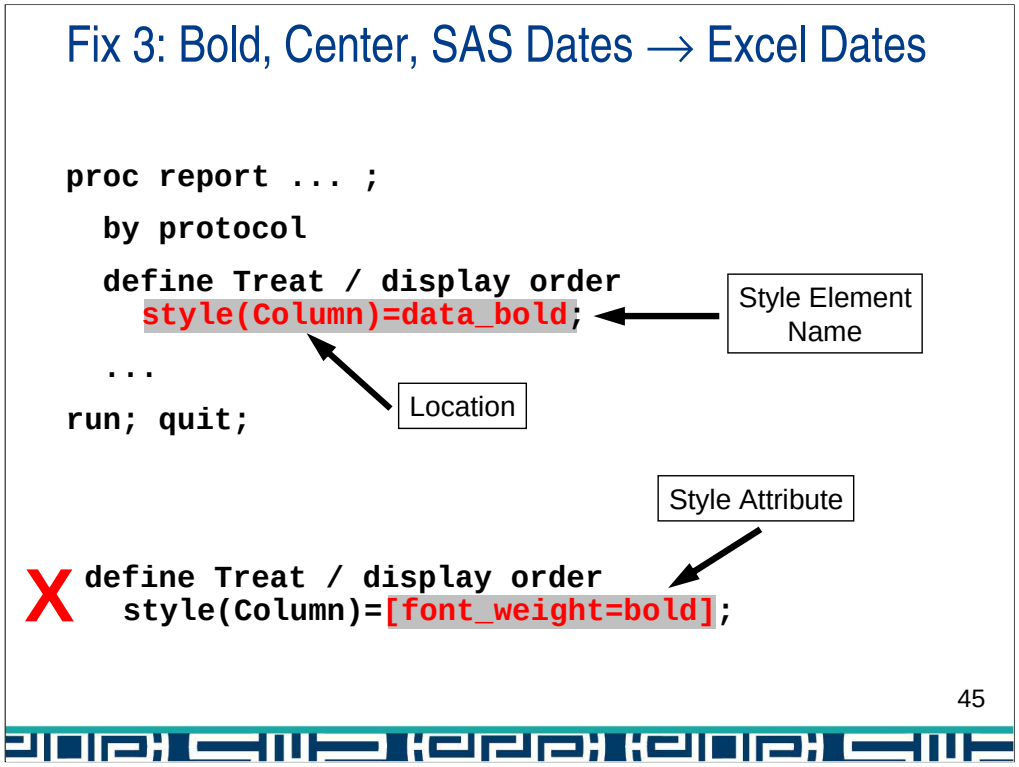


Recall that style overrides are used to apply style attributes or elements to specific parts of SAS output called *locations*. Here you can see the locations that are pertinent to the PROC REPORT output.

To change the appearance of the data cells in the "Treat" column, we apply a style override to the "Column" location.

Reference:

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



Adding this style override causes the values in the data cells for the "Treat" column to be displayed with bold text.

The style element name, which contains the style attribute setting, is used

```
style(Column)=data_bold
```

instead of the less-preferred format, which uses the style attribute setting

```
style(Column)=[font_weight=bold]
```

(Slide 17 shows the syntax for style overrides.)

Fix 3: Bold, Center, SAS Dates → Excel Dates

User-defined style element in XLsansPrinter style:

```
style data_center from data /
  just = center;
```

46

The XLsansPrinter style contains a style element named "data_center". It inherits all the attributes of the "data" style element that is supplied by SAS, but centers text within the cells.

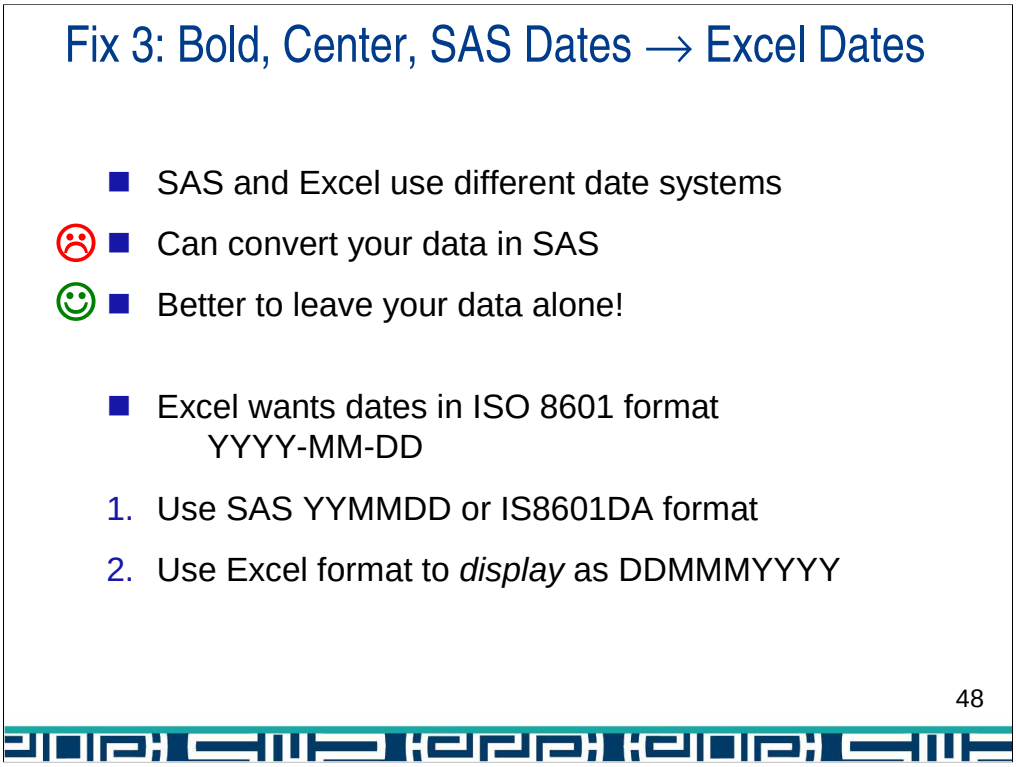
The "data_center" style element is applied to the "Patient", "LabSite", and "Visit" columns using a style override.

Fix 3: Bold, Center, SAS Dates → Excel Dates

```
define Patient / display order
  style(Column)=data_center;

define LabSite / display order
  style(Column)=data_center;

define Visit  / display order
  style(Column)=data_center;
```

47

Adding this style override causes the text in the data cells for the "Patient", "LabSite", and "Visit" columns to be centered.



SAS and Microsoft Excel use different date systems. Consequently, you often encounter problems when Excel reads SAS output containing date values.

One way to correct this behavior is to write SAS code to convert SAS date values to Excel date values, but this approach is problematic because you must alter your original SAS data. While you can create a new SAS view or table that contains the new date values, this is a vector for errors and becomes inefficient as your data grows.

A better solution, one that does not require you to alter the underlying data, is to use a combination of SAS and Excel formats. First you specify a SAS format using a FORMAT statement, and then you specify an Excel format using a style override. The SAS format changes what is physically written into the XML file, and the Excel format changes the way the value is displayed.

References:

SAS 9.1 – IS8601DA Format:
http://support.sas.com/kb/11/206.html

SAS 9.2 – E8601DA Format:
http://support.sas.com/documentation/cdl/en/lrdict/61724/HTML/default/a003169814.htm

Fix 3: Bold, Center, SAS Dates → Excel Dates

User-defined style element in XLsansPrinter style:

```
style data_date9 from data /
  tagattr='type:DateTime format:ddmmmyyyy';
```

Excel Format	Display Value
ddmmmyyyy	09Jun1997
m/d/yyyy	6/9/1997
mm/d/yyyy	06/9/1997
mm/dd/yyyy	06/09/1997
mmm d, yyyy	Jun 9, 1997
mmmm, d, yyyy	June 9, 1997
m/d/yy	6/9/97

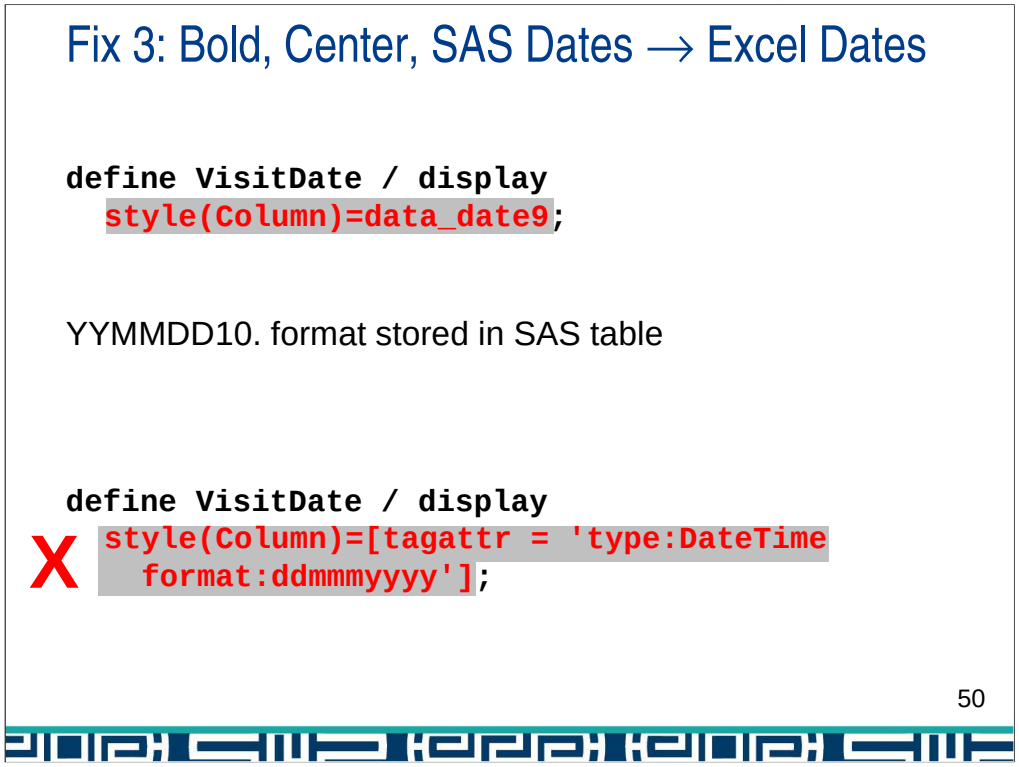
49

The "XLsansPrinter" style contains the style element "data_date9", which inherits all the attributes of the "data" style element that is supplied by SAS. The "tagattr" attribute is used to apply an Excel format to the data, and also to instruct Excel to interpret the value as an *Excel* datetime value.

This table shows how the date June 9, 1997 is displayed in Excel using different formats, including the format used by the "data_date9" style element ("ddmmmyyyy").

Reference:

To find out more about Excel date formats, type "display numbers as dates or times" into the Excel help system or refer to a book that covers this topic.



Because Excel expects dates to be represented using the ISO 8601 format (YYYY-MM-DD), SAS date values should be formatted using the YYMMDD or IS8601DA formats. The YYMMDD10. format was applied to the "VisitDate" column when the SAS table was created, providing Excel with dates in the correct format. No additional FORMAT statement is needed in this case.

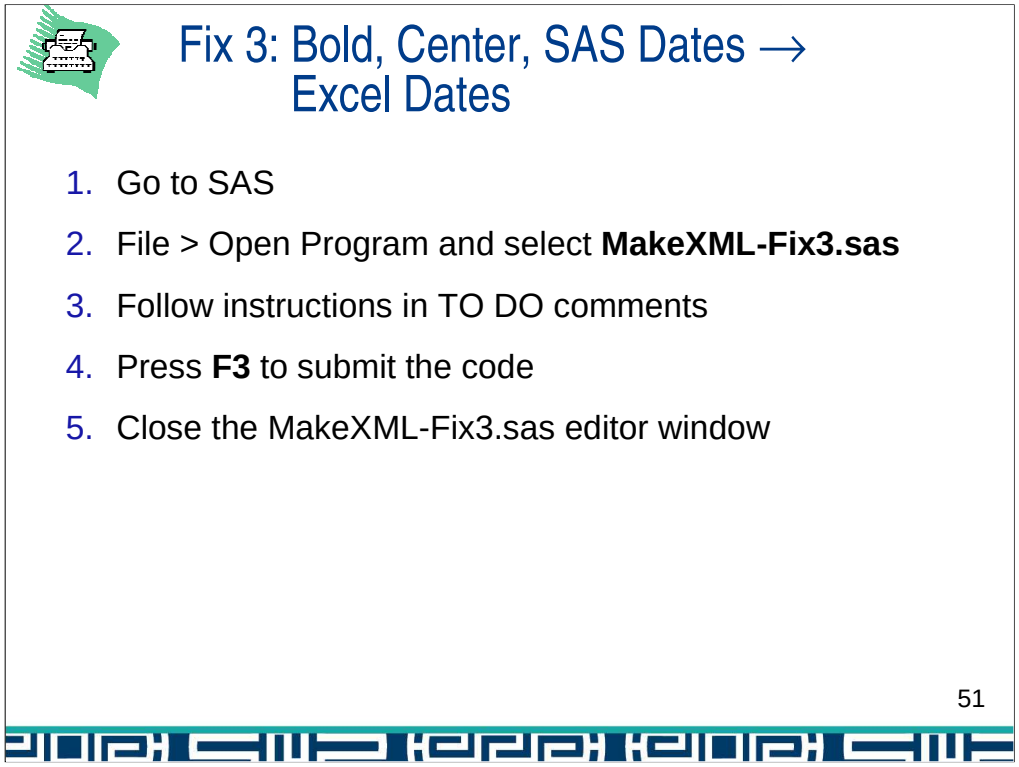
The Excel format, applied as a result of using the "data_date9" style element, causes the date values to be *displayed* according to the format DDMMMYYYY, instead of YYYY-MM-DD.

The style element name, which contains the style attribute settings, is used

```
style(Column)=data_date9
```

instead of the less-preferred format, which uses the style attribute settings

```
style(Column)=[tagattr='type:DateTime
  format:ddmmmyyyy]
```



TO DO:

Line 65: Note the style override applied to the Treat column to cause the text to be bold.

Line 66-68: Specify the name of the style element to the Patient, LabSite and Visit columns to cause the text to be centered.

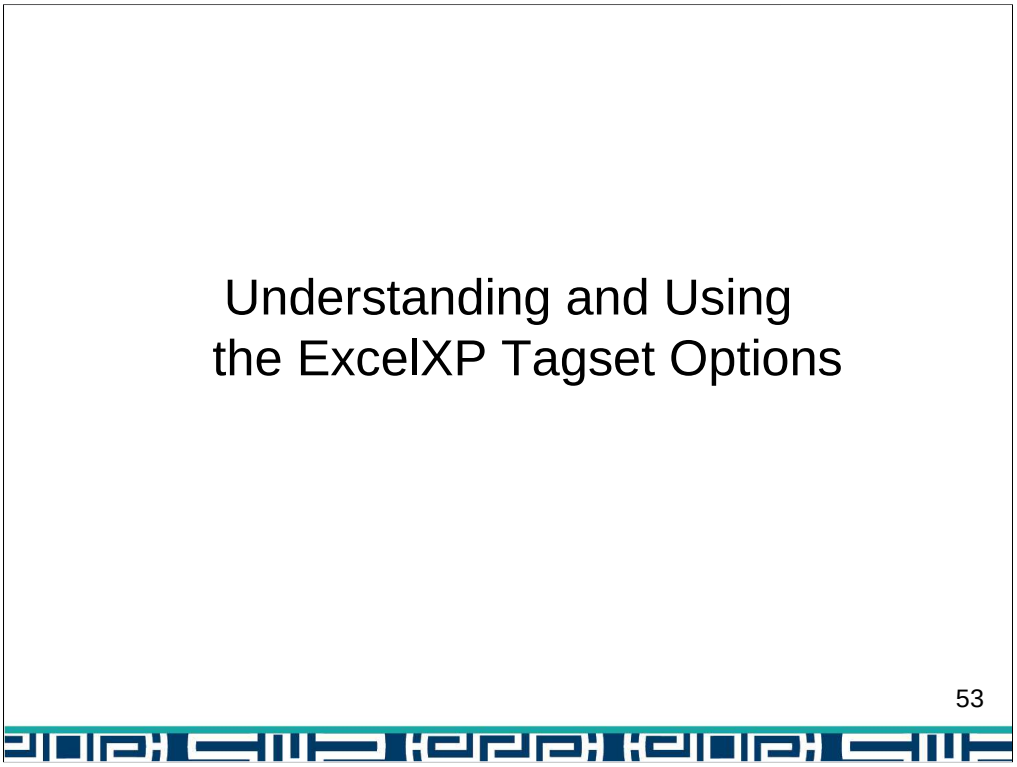
Line 69: Specify the name of the style element that applies the Excel format comparable to the SAS DATE9. format.

Solution:

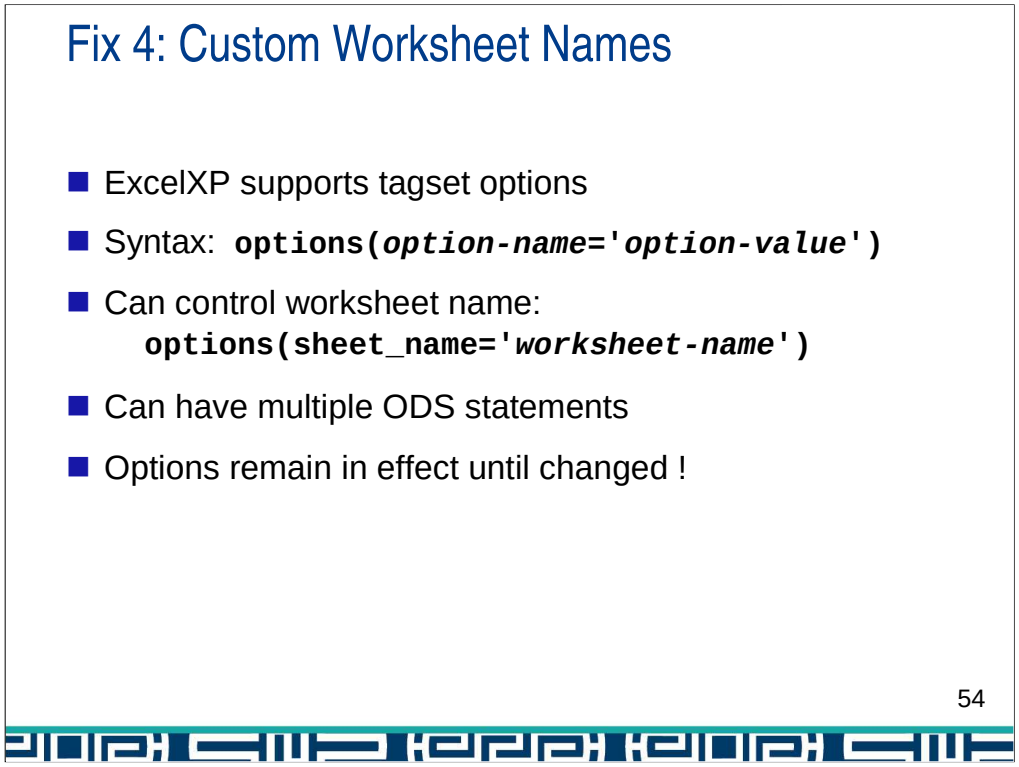
```
define Patient   / ... style(Column)=data_center;
define LabSite   / ... style(Column)=data_center;
define Visit     / ... style(Column)=data_center;
define VisitDate / ... style(Column)=data_date9;
```

Fix 3: Bold, Center, SAS Dates → Excel Dates

1. Go to Excel
2. File > **\HOW\DelGobbo\LabReport.xml**
   - or -
   LabReport.xml (from the recent file list)
3. Note formatting and dates
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. We use this method to name the range flags worksheet, but use a different technique to name the lab results worksheets.

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

Fix 4: Custom Worksheet Names

```
ods tagsets.ExcelXP style=XLsansPrinter ... ;
...
ods tagsets.ExcelXP options(sheet_name='Range Flags');

proc print ... ;

ods tagsets.ExcelXP options(sheet_name='none');

proc report ...;

ods tagsets.ExcelXP close;
```

55

Specify the SHEET_NAME option prior to the procedure code that creates the worksheet.

We reset the value of the option to "none" because we do not want the option setting ("Range Flags") to affect subsequent worksheets.

Fix 4: Custom Worksheet Names

```
ods tagsets.ExcelXP style=XLsansPrinter ... ;
...
ods tagsets.ExcelXP options(sheet_name='Range Flags');

proc print ... ;

ods tagsets.ExcelXP options(sheet_name='none');

ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label='16.2');

proc report ...;

ods tagsets.ExcelXP close;
```

56

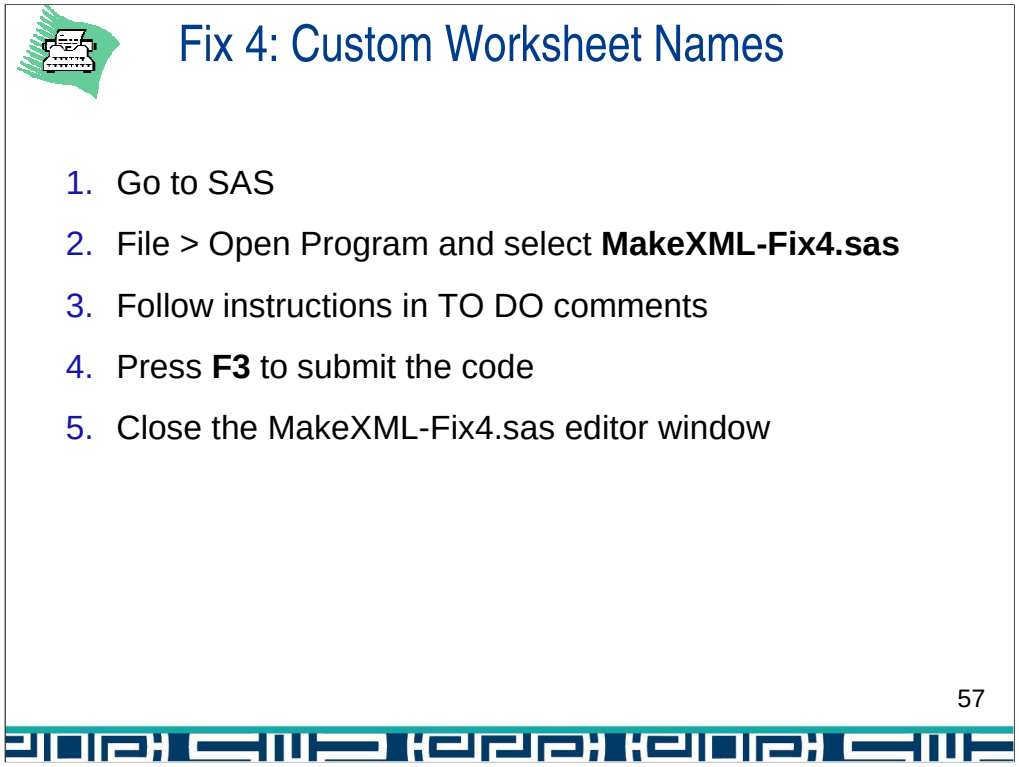
The SHEET_INTERVAL option controls the interval at which SAS output is placed into worksheets, and the SHEET_LABEL option is used to specify the prefix to use for the worksheet names.

When used together as shown here, the current value of the first BY group variable is prefixed with "16.2", and used as the worksheet name.

The BY variable (Protocol) in our REPORT procedure code contains 2 distinct values, "ABC 123" and "XYZ 987", resulting in worksheets named "16.2 ABC 123" and "16.2 XYZ 987".

The worksheet names are based on the International Committee on Harmonisation (ICH) Guideline E6 for clinical study report submissions. The guideline details the content and order of data collected in clinical trials. Patient data listings are found in section 16.2; hence, the naming of the worksheets.

If you do not want any text to prefix the BY value, specify `sheet_label=' '` (note the blank space between the quotation marks).



TO DO:

Line 26: Specify the name to use for the range flags worksheet.

Line 35: Note that the SHEET_NAME option has been reset to prevent unexpected worksheet names.

Line 39: Specify the text to precede the BY group values.

Solution:

```
ods tagsets.ExcelXP options(sheet_name='Range Flags');
ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label='16.2');
```

Fix 4: Custom Worksheet Names

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

58

Fix 5: Suppress BY Line Text

- BY line text before REPORT output
- Redundant – Included in worksheet names
- Use the SUPPRESS_BYLINES ***tagset*** option
- DO NOT use the NOBYLINE ***system*** option

59

BY line text appears in the lab results worksheets because the REPORT procedure is executed with a BY statement. The text is obscured because the first column is narrow, but it is also 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.

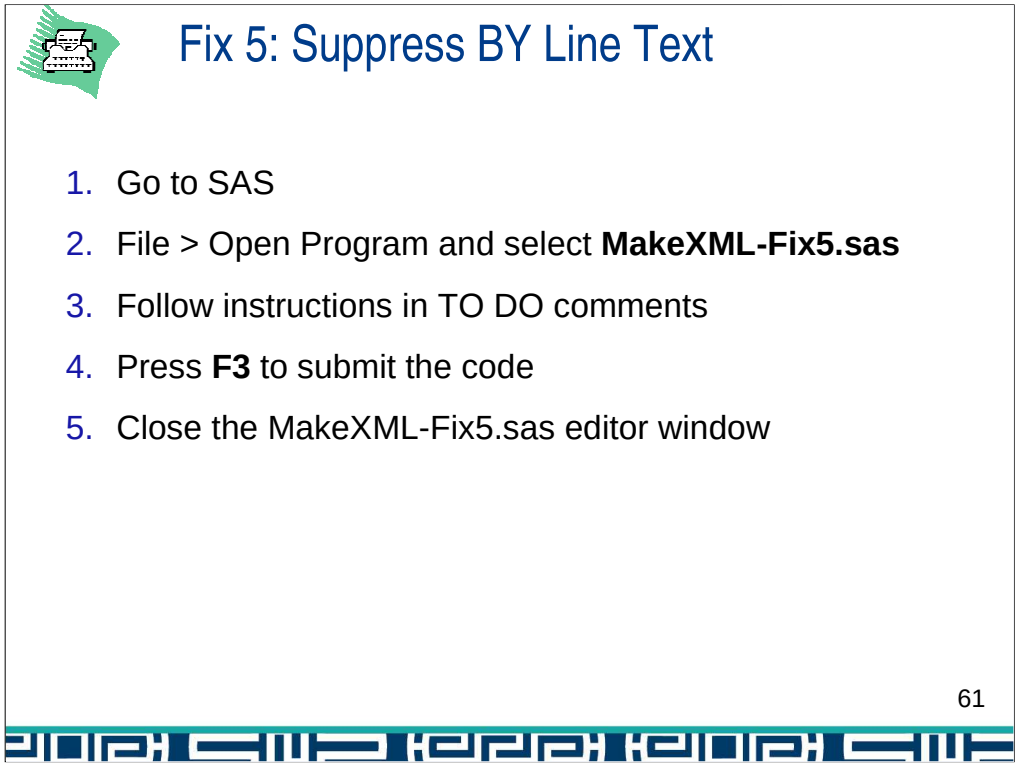
Do not attempt to use the NOBYLINE *system* option, as this disables BY group processing in the ExcelXP tagset when the SHEET_INTERVAL option is set to "bygroup".

Fix 5: Suppress BY Line Text

```
ods tagsets.ExcelXP style=XLsansPrinter ... ;
...
ods tagsets.ExcelXP options(sheet_name='Range Flags');

proc print ... ;

ods tagsets.ExcelXP options(sheet_name='none');

ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label='16.2'
                            suppress_bylines='yes');

proc report ...;

ods tagsets.ExcelXP close;
```

60

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



TO DO:

Line 36: Specify the tagset option to suppress the BY line text.

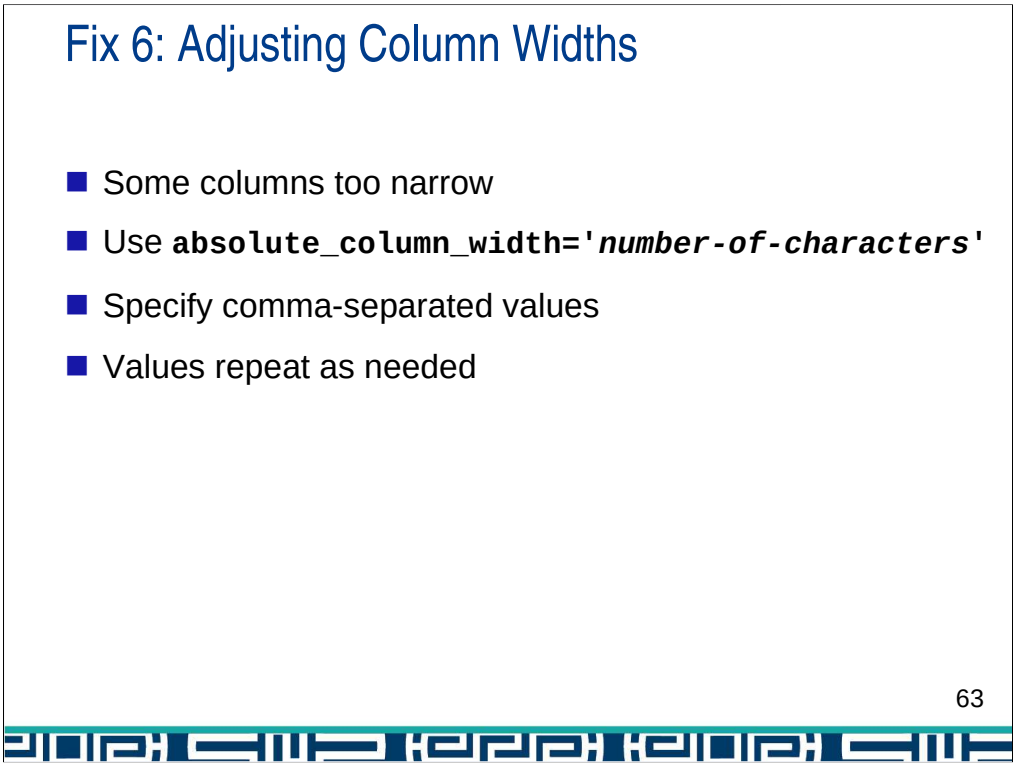
Solution:

```
ods tagsets.ExcelXP options(sheet_interval='bygroup'
     sheet_label='16.2'
     suppress_bylines='yes');
```

Fix 5: Suppress BY Line Text

1. Go to Excel
2. File > **\HOW\DelGobbo\LabReport.xml**
   - or -
   LabReport.xml (from the recent file list)
3. Note BY values are not in worksheet
4. Close the document (child) window (leave Excel running, but minimize it)

62



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, 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.

Fix 6: 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

64

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

Fix 6: Adjusting Column Widths

First guess:

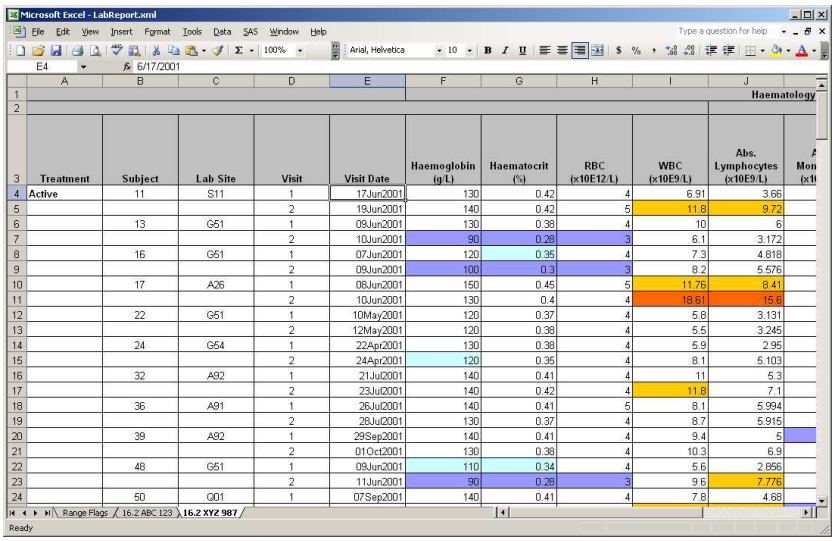
```
ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label='16.2'
                            suppress_bylines='yes'
                            absolute_column_width='10');
```

65

To approximate the values needed for the option, we start by specifying an arbitrary value of "10", which causes all of the columns to be the same size.

Fix 6: Adjusting Column Widths

Have:



Treatment	Subject	Lab Site	Visit	Visit Date	Haemoglobin (g/L)	Haematocrit (%)	RBC (x10E12/L)	WBC (x10E9/L)	Abs. Lymphocytes (x10E9/L)
Active	11	S11	1	17Jun2001	130	0.42	4	6.91	3.66
			2	19Jun2001	140	0.42	5	11.8	9.72
	13	G51	1	09Jun2001	130	0.38	4	10	6
			2	10Jun2001	90	0.28	3	6.1	3.172
	16	G51	1	07Jun2001	120	0.35	4	7.3	4.818
			2	09Jun2001	100	0.3	3	8.2	5.576
	17	A26	1	08Jun2001	150	0.45	5	11.76	8.41
			2	10Jun2001	130	0.4	4	18.61	15.6
	22	G51	1	10May2001	120	0.37	4	5.8	3.131
			2	12May2001	120	0.38	4	5.5	3.245
	24	G54	1	22Apr2001	130	0.38	4	5.9	2.95
			2	24Apr2001	120	0.35	4	8.1	5.103
	32	A92	1	21Jul2001	140	0.41	4	11	5.3
			2	23Jul2001	140	0.42	4	11.8	7.1
	36	A91	1	26Jul2001	140	0.41	5	8.1	5.994
			2	28Jul2001	130	0.37	4	8.7	5.915
	39	A92	1	29Sep2001	140	0.41	4	9.4	5
			2	01Oct2001	130	0.38	4	10.3	6.9
	48	G51	1	09Jun2001	110	0.34	4	5.6	2.856
			2	11Jun2001	90	0.28	3	9.6	7.776
	50	G01	1	07Sep2001	140	0.41	4	7.8	4.68

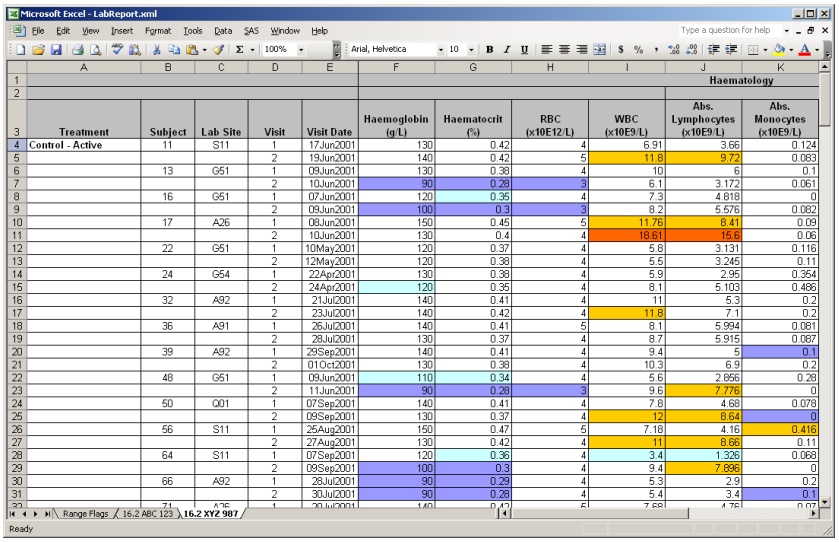
66

The width of all the columns containing the lab result values are reasonable, but data in the "Treatment" column is still obscured, and the "Subject", "Lab Site", "Visit", and "Visit Date" columns are all wider than they need to be.

Also, the column headings are too tall. This is corrected using the AUTOFIT_HEIGHT option.

Fix 6: Adjusting Column Widths

Want:



						Haematology					
	Treatment	Subject	Lab Site	Visit	Visit Date	Haemoglobin (g/L)	Haematocrit (%)	RBC (x10E12/L)	WBC (x10E9/L)	Abs. Lymphocytes (x10E9/L)	Abs. Monocytes (x10E9/L)
4	Control - Active	11	S11	1	17Jun2001	130	0.42	4	6.91	3.66	0.124
5				2	19Jun2001	140	0.42	5	11.8	9.72	0.083
6		13	G51	1	09Jun2001	130	0.38	4	10	6	0.1
7				2	10Jun2001	90	0.28	3	6.1	3.172	0.061
8		16	G51	1	07Jun2001	120	0.35	4	7.3	4.818	0
9				2	09Jun2001	100	0.3	3	8.2	5.576	0.082
10		17	A26	1	08Jun2001	150	0.45	5	11.76	8.41	0.09
11				2	10Jun2001	130	0.4	4	18.61	15.6	0.06
12		22	G51	1	10May2001	120	0.37	4	5.8	3.131	0.116
13				2	12May2001	120	0.38	4	5.5	3.245	0.11
14		24	G54	1	22Apr2001	130	0.38	4	5.9	2.95	0.354
15				2	24Apr2001	120	0.35	4	8.1	5.103	0.486
16		32	A92	1	21Jul2001	140	0.41	4	11	5.3	0.2
17				2	23Jul2001	140	0.42	4	11.8	7.1	0.2
18		36	A91	1	26Jul2001	140	0.41	5	8.1	5.994	0.081
19				2	28Jul2001	130	0.37	4	8.7	5.915	0.087
20		39	A92	1	29Sep2001	140	0.41	4	9.4	5	0.1
21				2	01Oct2001	130	0.38	4	10.3	6.9	0.2
22		48	G51	1	09Jun2001	110	0.34	4	5.6	2.856	0.28
23				2	11Jun2001	90	0.28	3	9.6	7.776	0
24		50	Q01	1	07Sep2001	140	0.41	4	7.8	4.68	0.078
25				2	09Sep2001	130	0.37	4	12	8.64	0
26		56	S11	1	25Aug2001	150	0.47	5	7.18	4.16	0.416
27				2	27Aug2001	130	0.42	4	11	8.66	0.11
28		64	S11	1	07Sep2001	120	0.36	4	3.4	1.326	0.068
29				2	09Sep2001	100	0.3	4	9.4	7.896	0
30		66	A92	1	28Jul2001	90	0.29	4	5.3	2.9	0.2
31				2	30Jul2001	90	0.28	4	5.4	3.4	0.1

The "Treatment" column is wider, the "Subject", "Lab Site", "Visit", and "Visit Date" columns are narrower, and the column headings are not too tall.

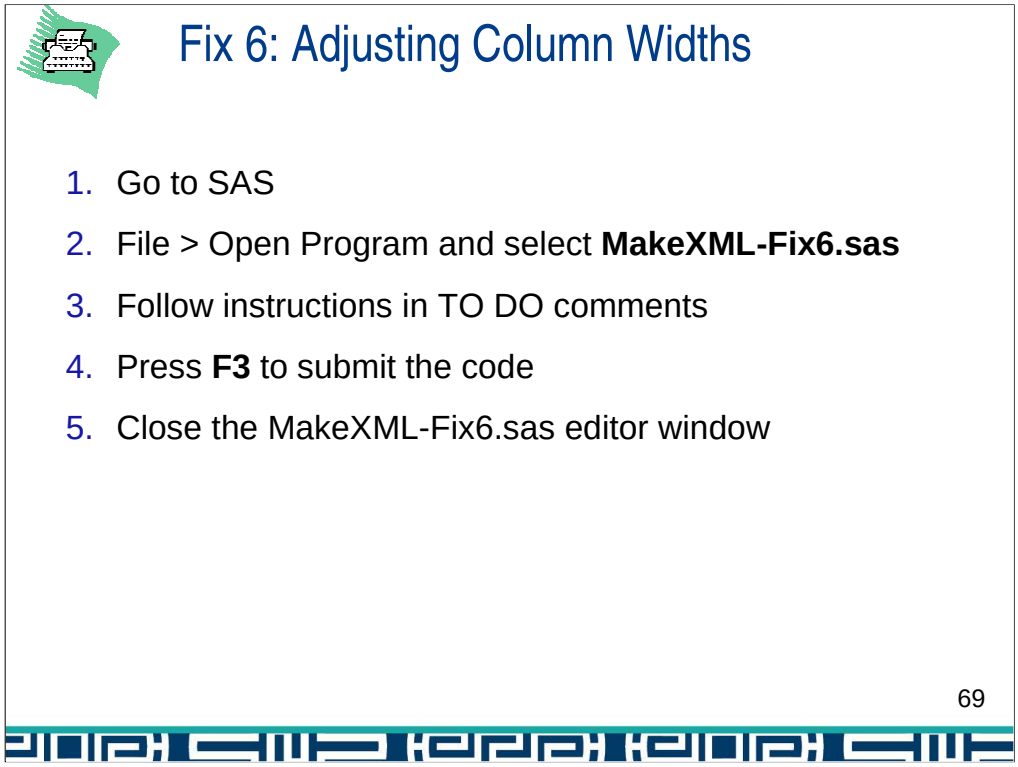
Fix 6: Adjusting Column Widths

```
ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label='16.2'
                            suppress_bylines='yes'
                            absolute_column_width='15,7,7,7,7,
                              10,10,10,10,10,10,10,10,10,10'
                            autofit_height='yes');
```

(ignore line wrapping above)

68

This code specifies more appropriate widths for each of the columns in the worksheet, and also corrects the row heights.



TO DO:

Line 39: Specify the tagset option to set the column widths.

Line 40: Note the AUTOFIT_HEIGHT option.

Solution (ignore line wrapping):

```
ods tagsets.ExcelXP options( ...
absolute_column_width='15,7,7,7,7,
10,10,10,10,10,10,10,10,10,10'
autofit_height='yes' );
```

Fix 6: Adjusting Column Widths

1. Go to Excel
2. File > **\HOW\DelGobbo\LabReport.xml**
   - or -
   LabReport.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)

70

Homework: Frozen Headings

- Column headings "lost" when scrolling
- Row headings "lost" when scrolling
- Tagset options are available

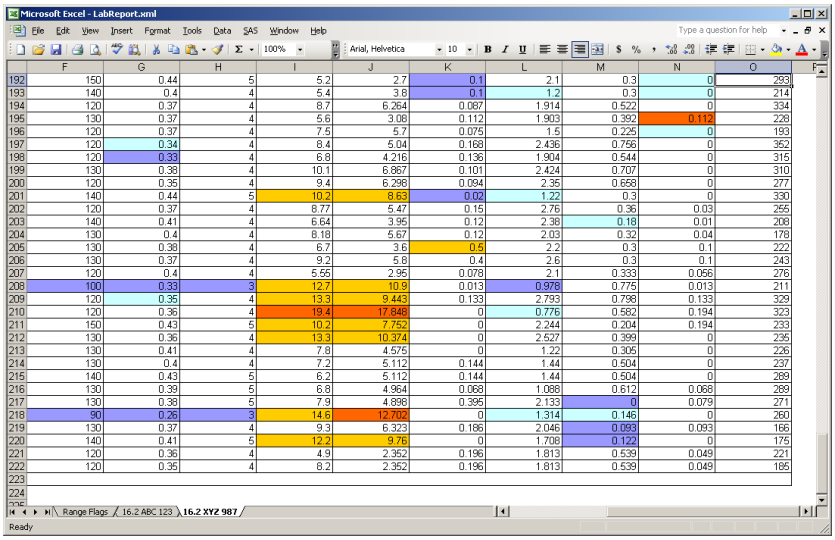
71

When you scroll down in a workbook that contains a lot of data, the column headings can scroll off the top of the viewable screen. You can correct this problem by using the FROZEN_HEADERS tagset option to "freeze" the rows you want to use as column headings.

Similarly, scrolling to the right causes the "Treatment", "Subject", "Lab Site", "Visit", and "Visit Date" columns to move off the left of the viewable screen. This problem is corrected by using the FROZEN_ROWHEADERS option.

Homework: Frozen Headings

Have:

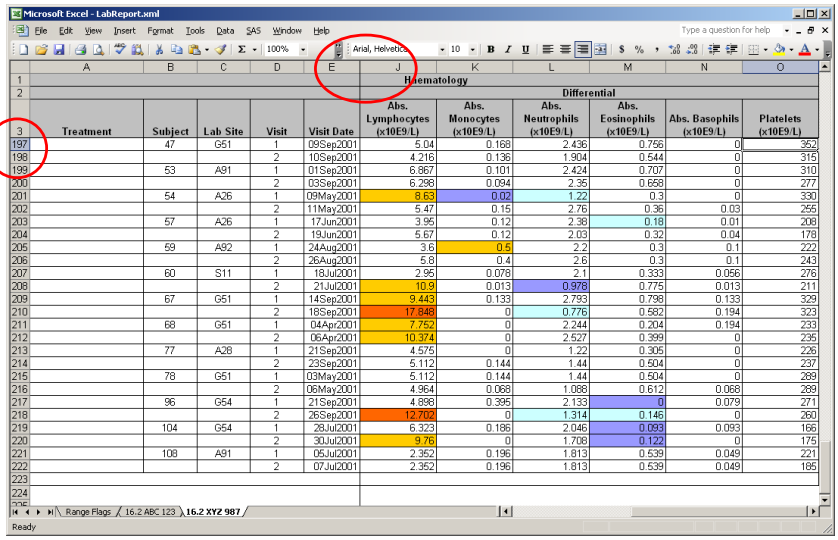


72

Column and row headings are not visible.

Homework: Frozen Headings

Want:



73

Visible column and row headings.

Homework: Frozen Headings

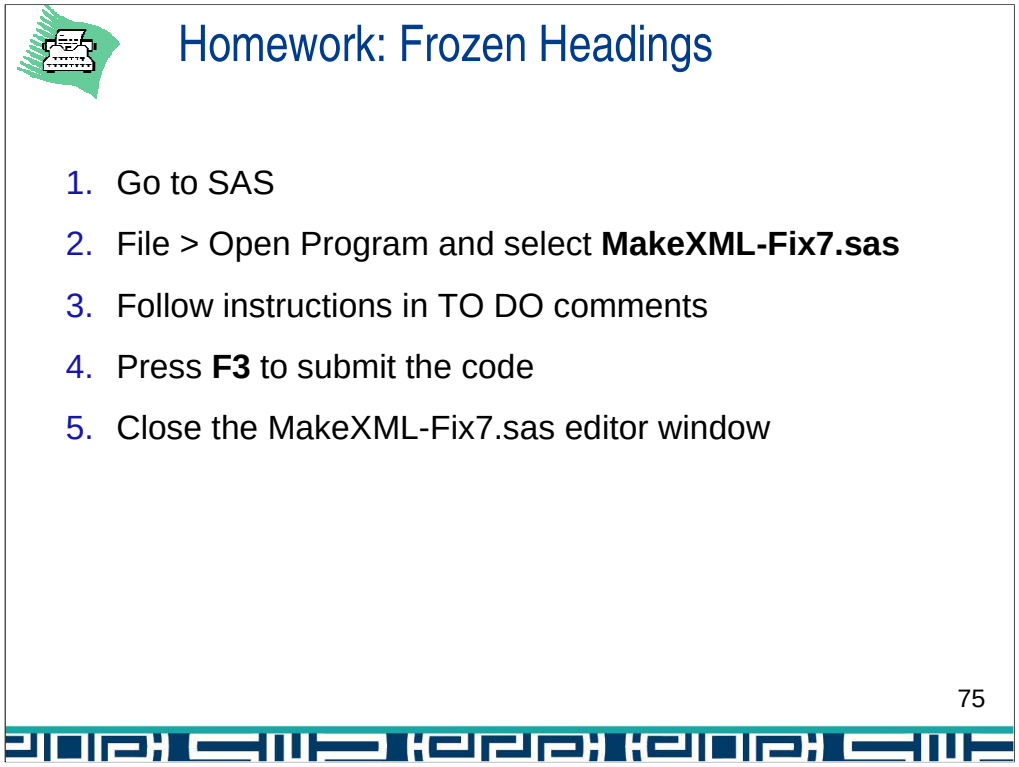
```
ods tagsets.ExcelXP options(sheet_interval='bygroup'
                            sheet_label='16.2'
                            suppress_bylines='yes'
                            absolute_column_width='...'
                            autofit_height='yes'
                            frozen_headers='yes'
                            frozen_rowheaders='5');
```

74

Valid values for both of these options are either "yes" or an integer. If you specify "yes", the tagset will attempt to automatically determine the columns or rows to freeze.

If you do not like the results, you can explicitly specify the rows or columns to freeze. For example, to freeze rows 1-3, specify `frozen_headers='3'`.

Specifying a value of "5" for FROZEN_ROWHEADERS ensures that Excel columns "A-E" are frozen when scrolling to the right.



TO DO:

Lines 39-40: Specify the tagset options to "freeze" the column and row headings.

Solution:

```
ods tagsets.ExcelXP options( ...
                             frozen_headers='yes'
                             frozen_rowheaders='5');
```

Homework: Frozen Headings

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

Summary: Creating Multi-Sheet Workbooks

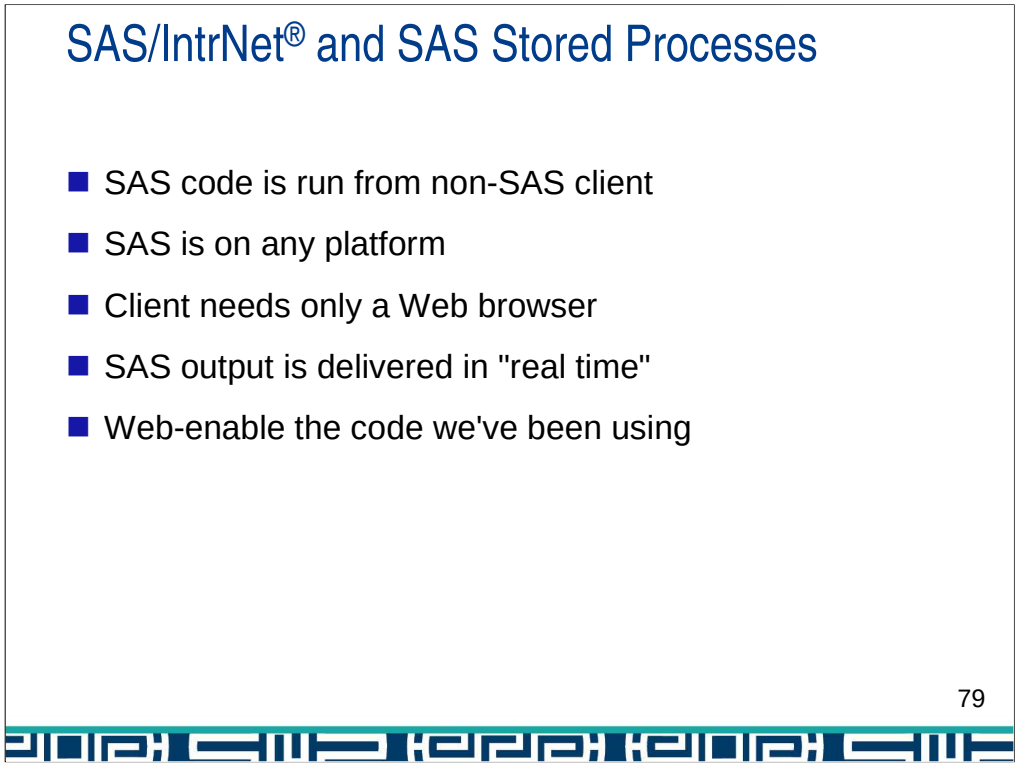
- Use ExcelXP tagset to create XML file
- Resulting XML file can be viewed with Excel
- Apply ODS style overrides carefully
- Make use of tagset options
- Use Excel formats instead of SAS formats

Using SAS/IntrNet®

and

SAS Stored Processes

78



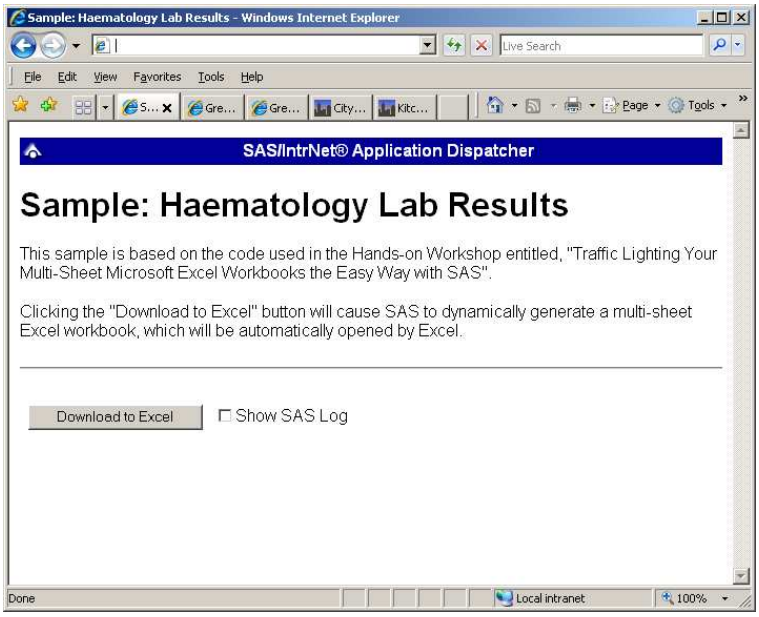
The purpose of the SAS/IntrNet® Application Dispatcher or SAS Stored Processes are to allow you to execute SAS programs from a client machine that does not have SAS installed. The client machine *may* have SAS installed, but that is not required.

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



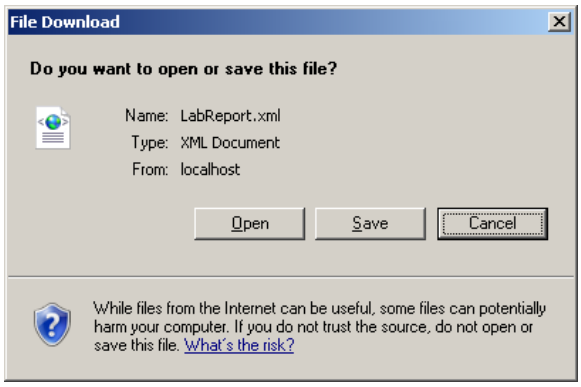
80

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



81

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

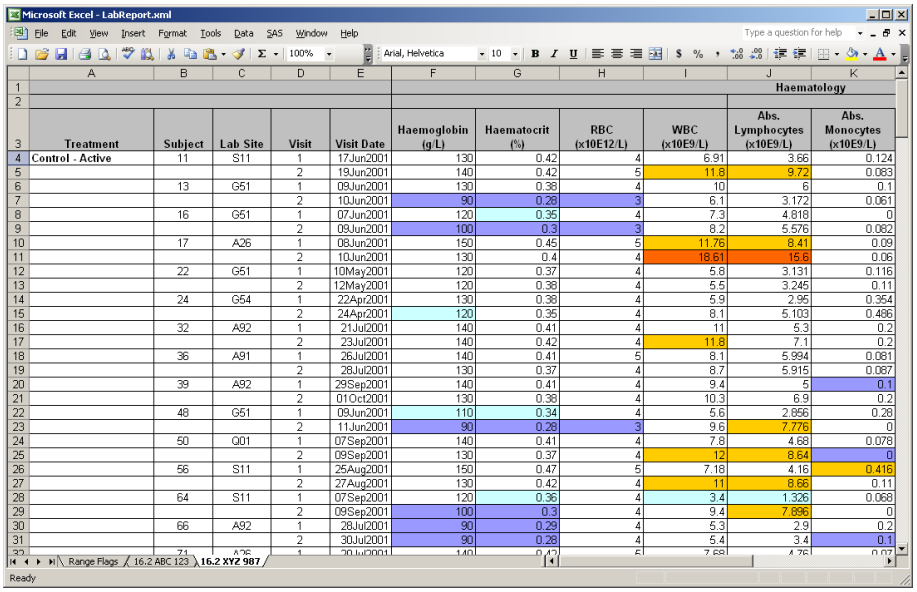
Note that you are presented with a "File Download" dialog box, instead of the results being displayed in the Web browser. Clicking "Open" causes the SAS output to be displayed in Excel, provided that Excel is installed on the client machine.

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



	A	B	C	D	E	F	G	H	I	J	K
1										Haematology	
2											
3	Treatment	Subject	Lab Site	Visit	Visit Date	Haemoglobin (g/L)	Haematocrit (%)	RBC (x10E12/L)	WBC (x10E9/L)	Abs. Lymphocytes (x10E9/L)	Abs. Monocytes (x10E9/L)
4	Control - Active	11	S11	1	17Jun2001	130	0.42	4	6.91	3.66	0.124
5				2	19Jun2001	140	0.42	5	11.8	9.72	0.083
6		13	G51	1	09Jun2001	130	0.38	4	10	6	0.1
7				2	10Jun2001	90	0.28	3	6.1	3.172	0.061
8		16	G51	1	07Jun2001	120	0.35	4	7.3	4.818	0
9				2	09Jun2001	100	0.3	3	8.2	5.576	0.082
10		17	A26	1	08Jun2001	150	0.45	5	11.76	8.41	0.09
11				2	10Jun2001	130	0.4	4	18.61	15.6	0.06
12		22	G51	1	10May2001	120	0.37	4	5.8	3.131	0.116
13				2	12May2001	120	0.38	4	5.5	3.245	0.11
14		24	G54	1	22Apr2001	130	0.36	4	5.9	2.95	0.354
15				2	24Apr2001	120	0.35	4	8.1	5.103	0.486
16		32	A92	1	21Jul2001	140	0.41	4	11	5.3	0.2
17				2	23Jul2001	140	0.42	4	11.8	7.1	0.2
18		36	A91	1	26Jul2001	140	0.41	5	8.1	5.994	0.081
19				2	28Jul2001	130	0.37	4	8.7	5.916	0.087
20		39	A92	1	29Sep2001	140	0.41	4	9.4	5	0.1
21				2	01Oct2001	130	0.38	4	10.3	6.9	0.2
22		48	G51	1	09Jun2001	110	0.34	4	5.6	2.856	0.28
23				2	11Jun2001	90	0.28	3	9.6	7.776	0
24		50	Q01	1	07Sep2001	140	0.41	4	7.8	4.68	0.078
25				2	09Sep2001	130	0.37	4	12	8.64	0
26		56	S11	1	25Aug2001	150	0.47	5	7.18	4.16	0.416
27				2	27Aug2001	130	0.42	4	11	8.66	0.11
28		64	S11	1	07Sep2001	120	0.36	4	3.4	1.326	0.068
29				2	09Sep2001	100	0.3	4	9.4	7.896	0
30		66	A92	1	28Jul2001	90	0.29	4	5.3	2.9	0.2
31				2	30Jul2001	90	0.28	4	5.4	3.4	0.1

82

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

Conclusion

- The ExcelXP tagset creates much sought-after multi-sheet workbooks from SAS output
- Use tagset options to increase functionality of workbooks
- SAS/IntrNet® and SAS Stored Processes can be used to deliver dynamic SAS output over an intranet or the Internet

83

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#excel2010

  support.sas.com/events/sasglobalforum/2010/handson

- ExcelXP Tagset Paper Index:

  www.sas.com/events/cm/867226/ExcelXPPaperIndex.pdf

84

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#excel2010 and find the entry "Traffic Lighting Your 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.

Contact Information

Please send questions, comments and feedback to:

Vince DelGobbo

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 (support.sas.com/usergroups/namerica/lug-form.html) at least eight weeks in advance.

85

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.