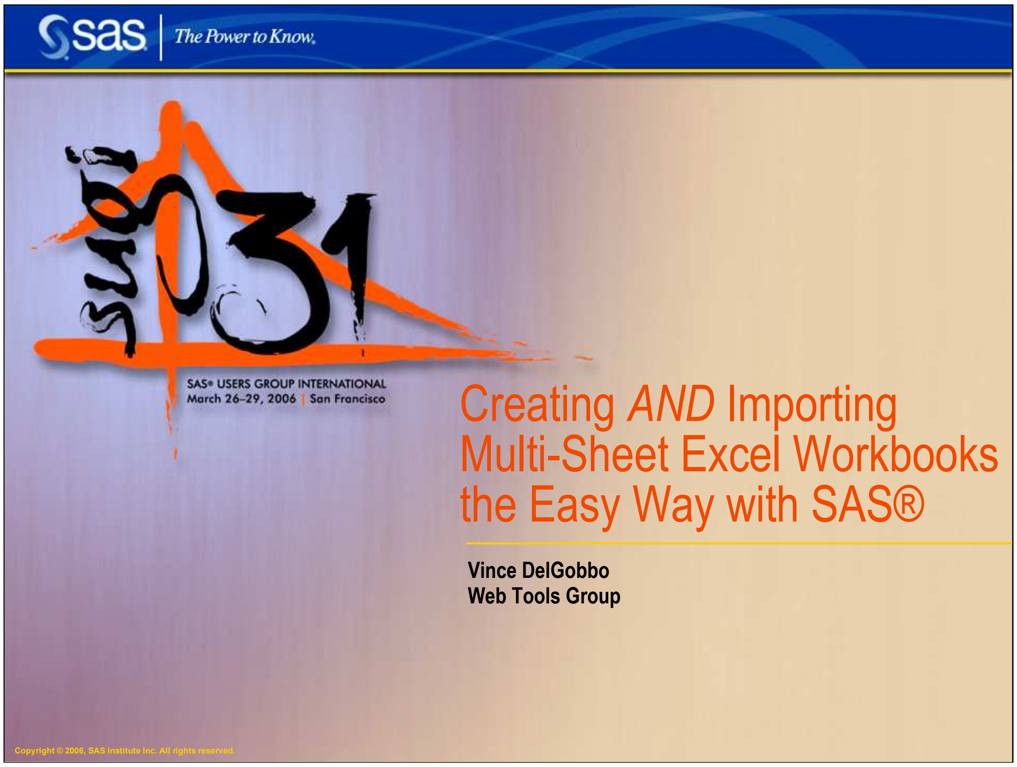


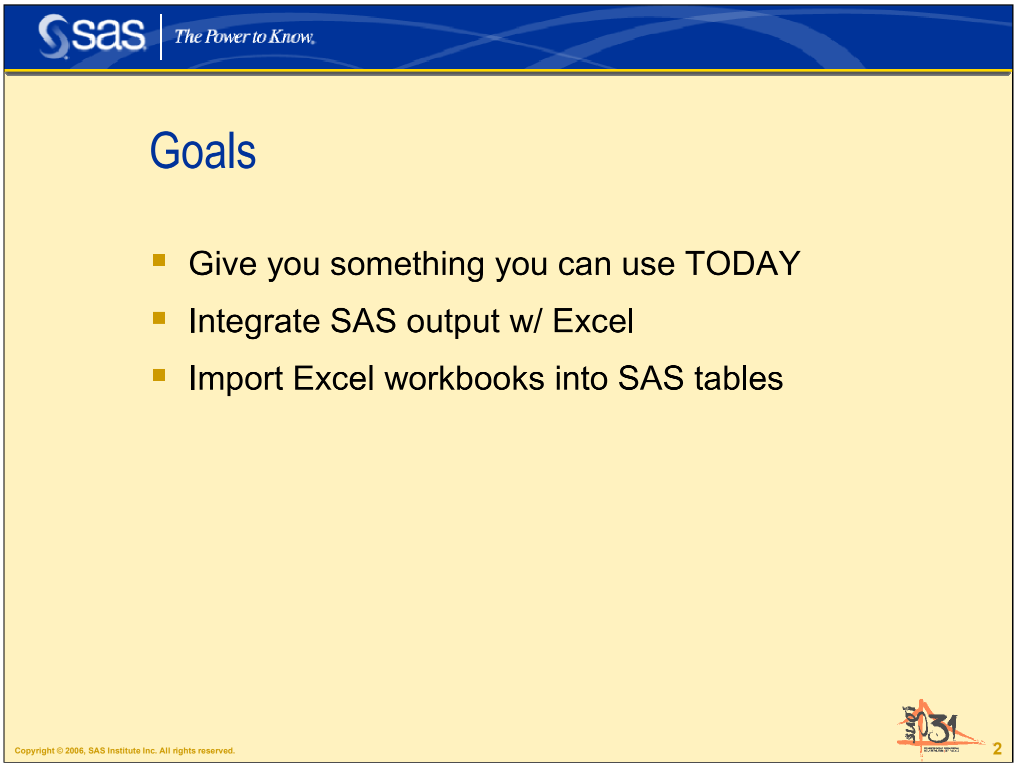
SAS® USERS GROUP INTERNATIONAL
March 26–29, 2006 | San Francisco

Creating *AND* Importing Multi-Sheet Excel Workbooks the Easy Way with SAS®

Vince DelGobbo
Web Tools Group

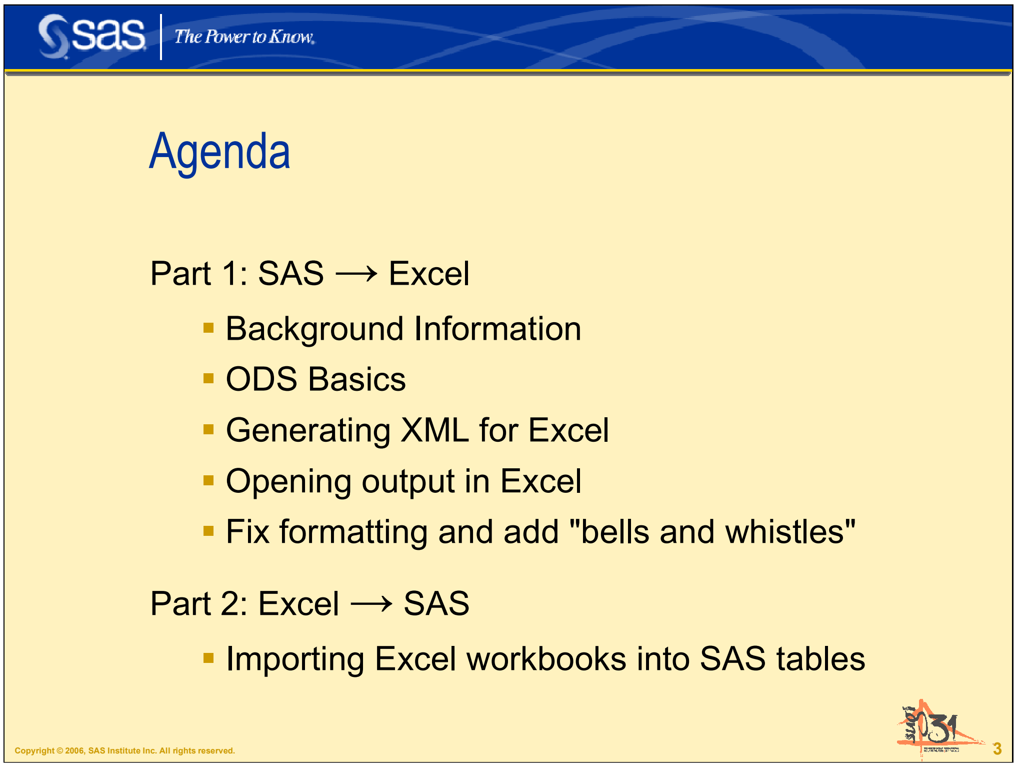
Presented at the 31st SAS Users Group International Conference (SUGI)
Moscone West Convention Center
San Francisco, CA
March 26-29, 2006

A special *"thank you"* to the SUGI Hands-on Workshop section chairs Lori Griffin and Derek Morgan for inviting me to present this topic, and to Chris Barrett of SAS Institute Inc. for his valuable input on the accompanying paper.



Goals

- Give you something you can use TODAY
- Integrate SAS output w/ Excel
- Import Excel workbooks into SAS tables



The Power to Know.

Agenda

Part 1: SAS ⟶ Excel

- Background Information
- ODS Basics
- Generating XML for Excel
- Opening output in Excel
- Fix formatting and add "bells and whistles"

Part 2: Excel ⟶ SAS

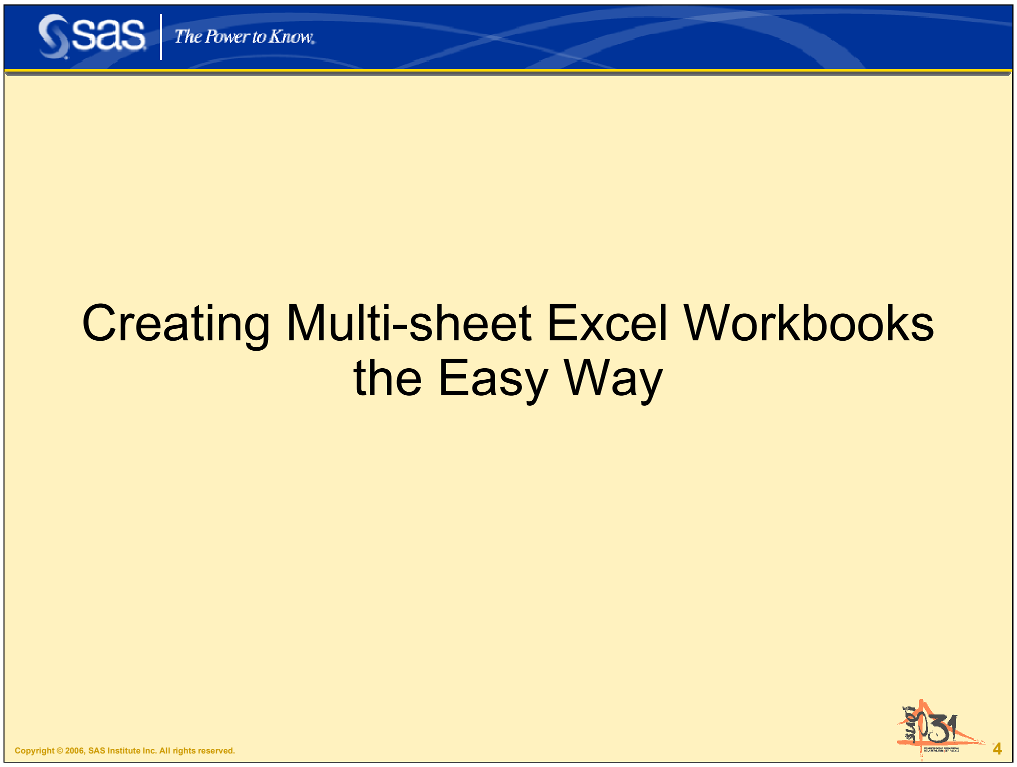
- Importing Excel workbooks into SAS tables

Software Requirements

- Base SAS, ***any*** operating system.

- SAS 9.1.2 or later (9.1.3 provides greatly improved performance for the "ExcelXP" tagset).

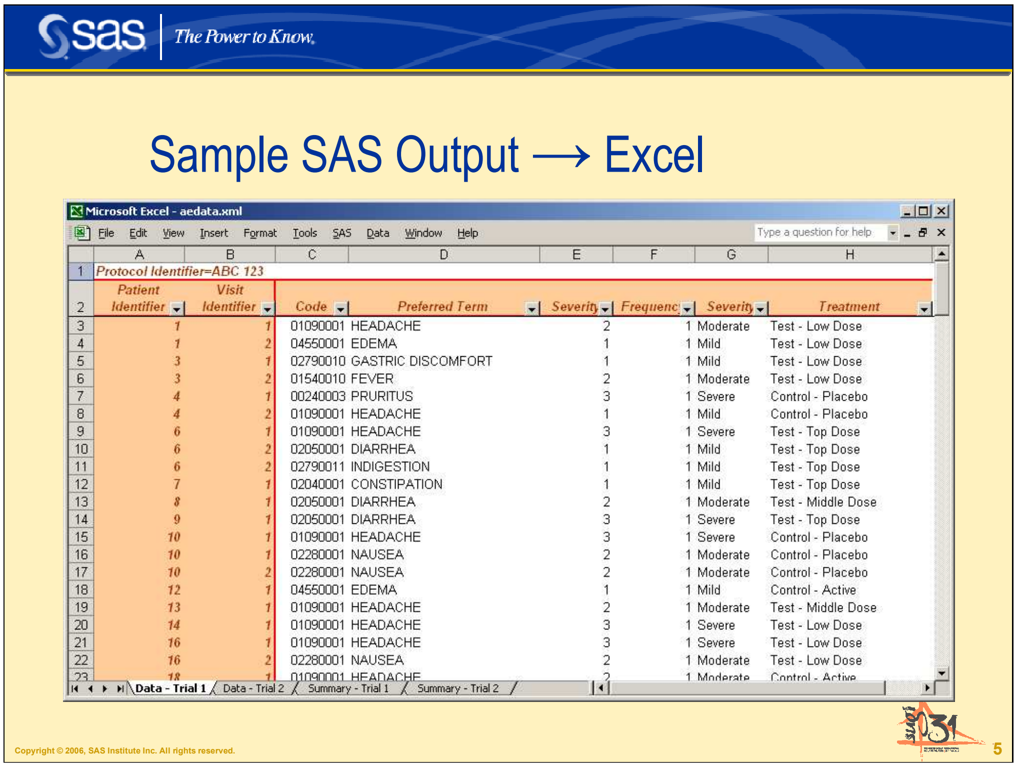
- Microsoft Excel XP (a.k.a. Excel 2002).



Creating Multi-sheet Excel Workbooks the Easy Way

This section focuses on using ODS to create XML files that can be imported into Excel.

A later section will focus on importing Excel workbooks into SAS tables.



Sample SAS Output ⟶ Excel

Protocol Identifier=ABC 123

	A	B	C	D	E	F	G	H
2	*Patient Identifier*	*Visit Identifier*	*Code*	*Preferred Term*	*Severity*	*Frequenc*	*Severity*	*Treatment*
3	1	1	01090001	HEADACHE	2	1	Moderate	Test - Low Dose
4	1	2	04550001	EDEMA	1	1	Mild	Test - Low Dose
5	3	1	02790010	GASTRIC DISCOMFORT	1	1	Mild	Test - Low Dose
6	3	2	01540010	FEVER	2	1	Moderate	Test - Low Dose
7	4	1	00240003	PRURITUS	3	1	Severe	Control - Placebo
8	4	2	01090001	HEADACHE	1	1	Mild	Control - Placebo
9	6	1	01090001	HEADACHE	3	1	Severe	Test - Top Dose
10	6	2	02050001	DIARRHEA	1	1	Mild	Test - Top Dose
11	6	2	02790011	INDIGESTION	1	1	Mild	Test - Top Dose
12	7	1	02040001	CONSTIPATION	1	1	Mild	Test - Top Dose
13	8	1	02050001	DIARRHEA	2	1	Moderate	Test - Middle Dose
14	9	1	02050001	DIARRHEA	3	1	Severe	Test - Top Dose
15	10	1	01090001	HEADACHE	3	1	Severe	Control - Placebo
16	10	1	02280001	NAUSEA	2	1	Moderate	Control - Placebo
17	10	2	02280001	NAUSEA	2	1	Moderate	Control - Placebo
18	12	1	04550001	EDEMA	1	1	Mild	Control - Active
19	13	1	01090001	HEADACHE	2	1	Moderate	Test - Middle Dose
20	14	1	01090001	HEADACHE	3	1	Severe	Test - Low Dose
21	16	1	01090001	HEADACHE	3	1	Severe	Test - Low Dose
22	16	2	02280001	NAUSEA	2	1	Moderate	Test - Low Dose
23	18	1	01090001	HEADACHE	2	1	Moderate	Control - Active

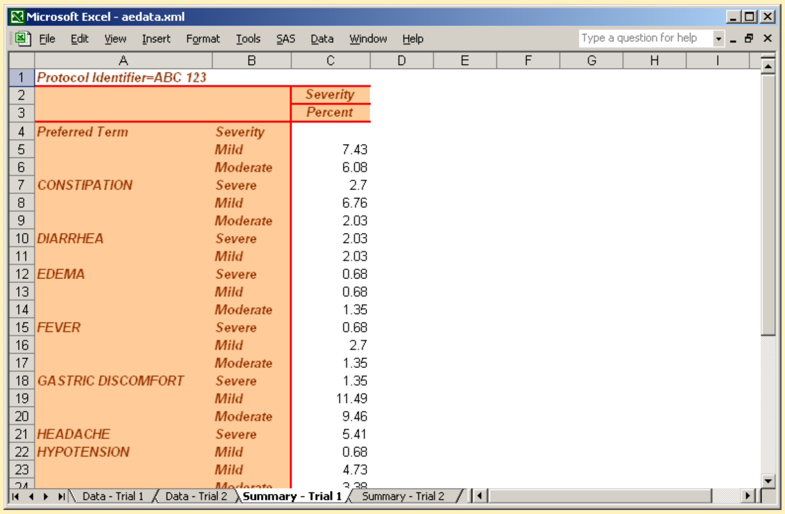
This illustrates the type of SAS output that we will be incorporating into Excel.

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

This Excel workbook has 4 worksheets, two created using the PRINT procedure, and two created using the TABULATE procedure. The visible worksheet represents output from the PRINT procedure.

Some things to note are the custom worksheet names, Excel AutoFilters, appropriate widths of columns and the color scheme, which resembles the SUGI 31 conference materials and Web site.

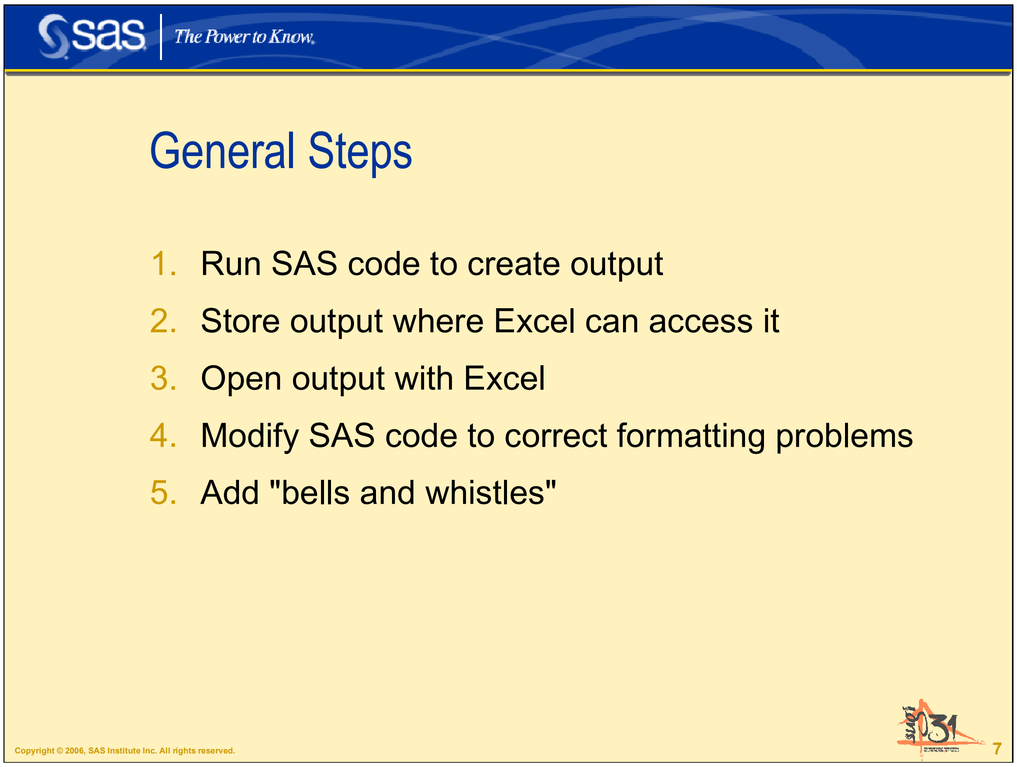
Sample SAS Output ⟶ Excel



Protocol Identifier=ABC 123

		Severity
		Percent
Preferred Term	Severity	
	Mild	7.43
	Moderate	6.08
CONSTIPATION	Severe	2.7
	Mild	6.76
	Moderate	2.03
DIARRHEA	Severe	2.03
	Mild	2.03
EDEMA	Severe	0.68
	Mild	0.68
	Moderate	1.35
FEVER	Severe	0.68
	Mild	2.7
	Moderate	1.35
GASTRIC DISCOMFORT	Severe	1.35
	Mild	11.49
	Moderate	9.46
HEADACHE	Severe	5.41
HYPOTENSION	Mild	0.68
	Mild	4.73
	Moderate	3.38

The visible worksheet represents output from the TABULATE procedure.



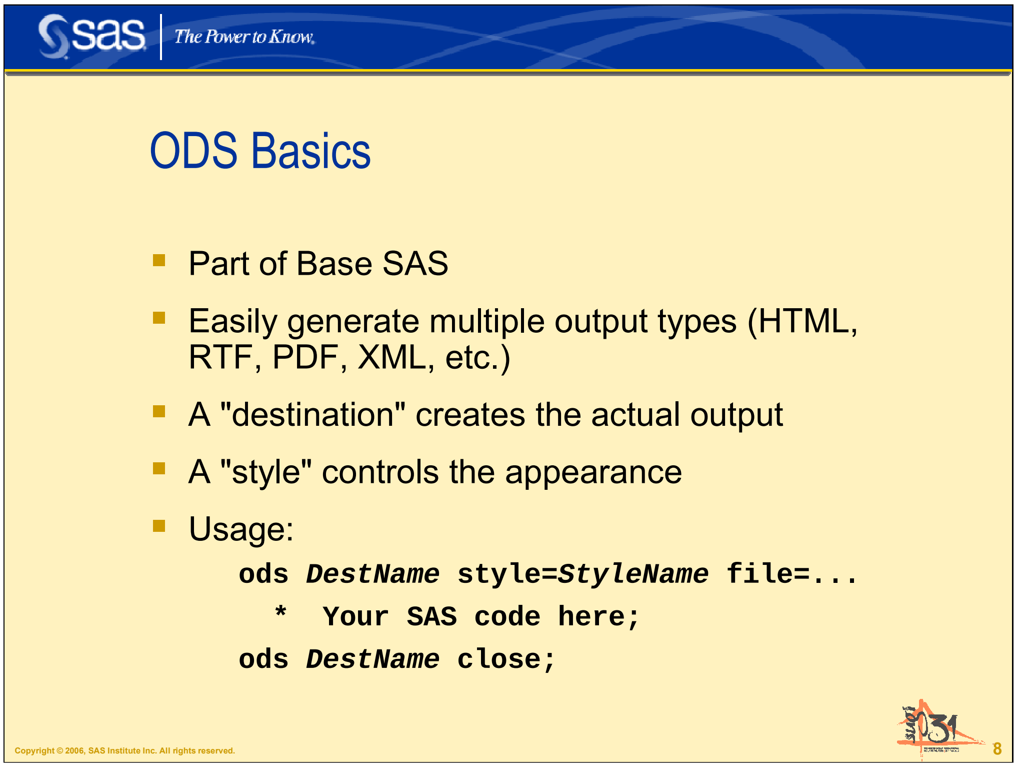
General Steps

1. Run SAS code to create output
2. Store output where Excel can access it
3. Open output with Excel
4. Modify SAS code to correct formatting problems
5. Add "bells and whistles"

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

Then the ODS output will be opened using Excel. If you have ever done this before, you have probably encountered formatting problems.

We will then explore techniques to instruct ODS to create output that Excel is happy with, and add some "bells and whistles", too.



ODS Basics

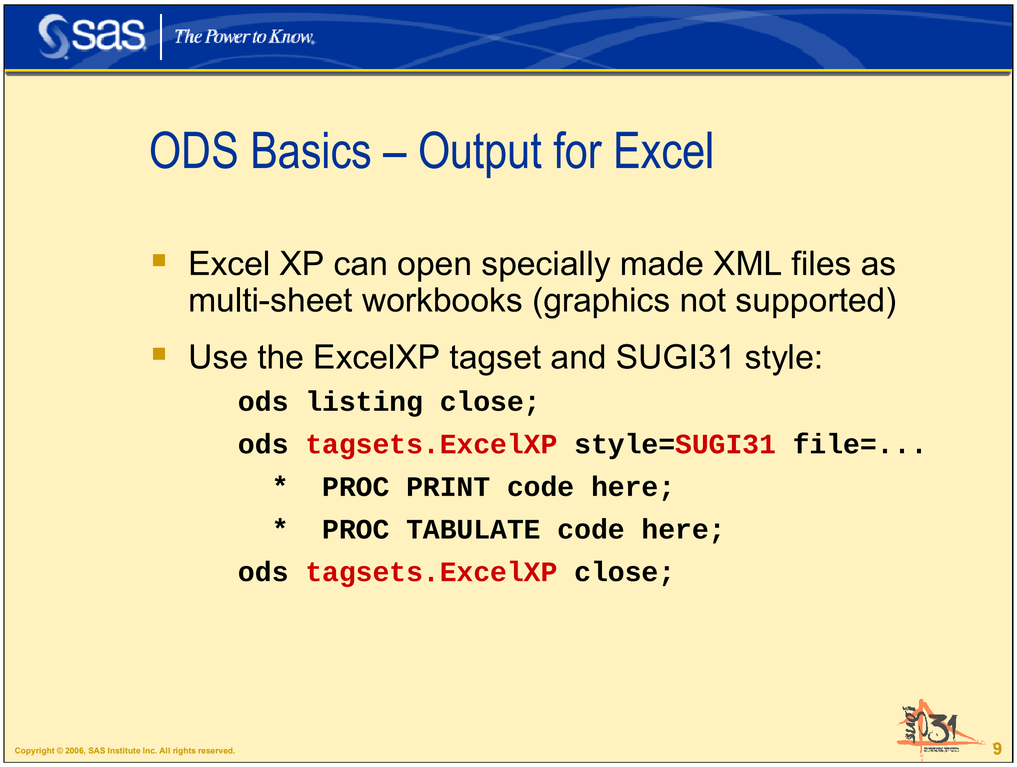
- Part of Base SAS
- Easily generate multiple output types (HTML, RTF, PDF, XML, etc.)
- A "destination" creates the actual output
- A "style" controls the appearance
- Usage:

```
ods DestName style=StyleName file=...
  *  Your SAS code here;
ods DestName close;
```

ODS destinations create the actual output (HTML, RTF, PDF, XML, etc.) while an ODS style controls the appearance (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" will be used.

In this workshop, we will be using a special 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.



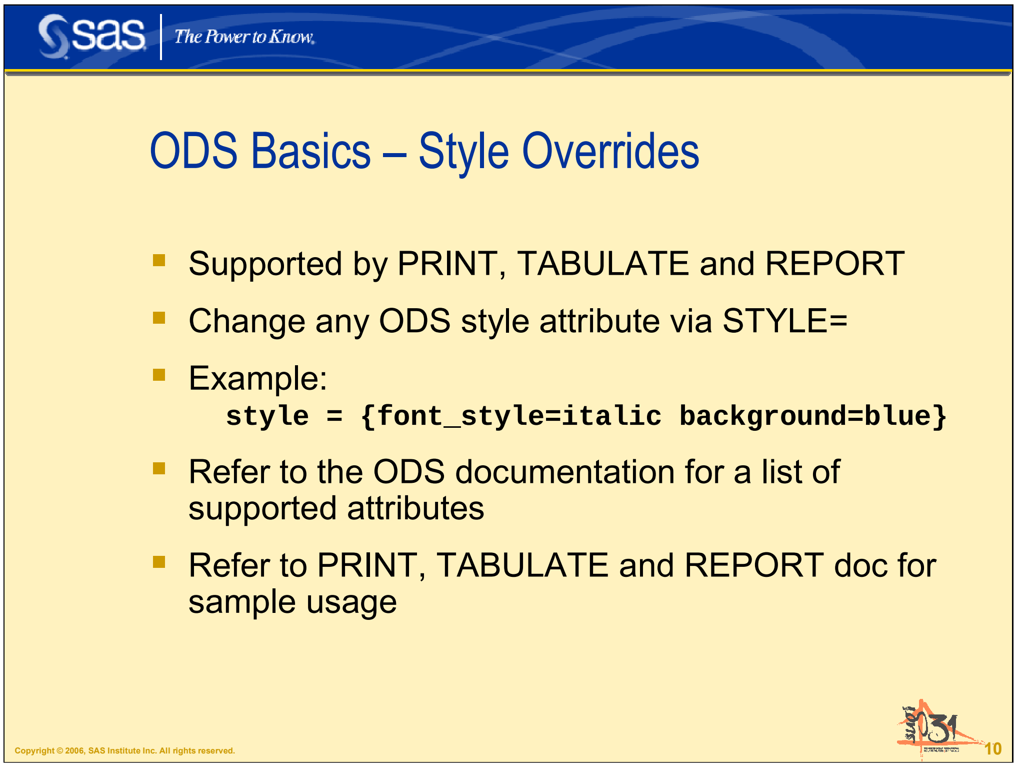
ODS Basics – Output for Excel

- Excel XP can open specially made XML files as multi-sheet workbooks (graphics not supported)
- Use the ExcelXP tagset and SUGI31 style:

```
ods listing close;
ods tagsets.ExcelXP style=SUGI31 file=...
  *  PROC PRINT code here;
  *  PROC TABULATE code here;
ods tagsets.ExcelXP close;
```

Currently, the Microsoft XML Spreadsheet Specification does not support graphics. Thus, SAS/GRAPH procedures are not supported. This is a Microsoft Excel limitation, not a limitation of the ExcelXP tagset.

We will use the ODS tagset named "ExcelXP" to create XML output that can be opened using Microsoft Excel XP. When opened with Excel, the XML file will be rendered as a multi-sheet Excel workbook. Additionally, we will use an ODS style named "SUGI31", to provide a look similar to the SUGI 31 conference materials and Web site.



ODS Basics – Style Overrides

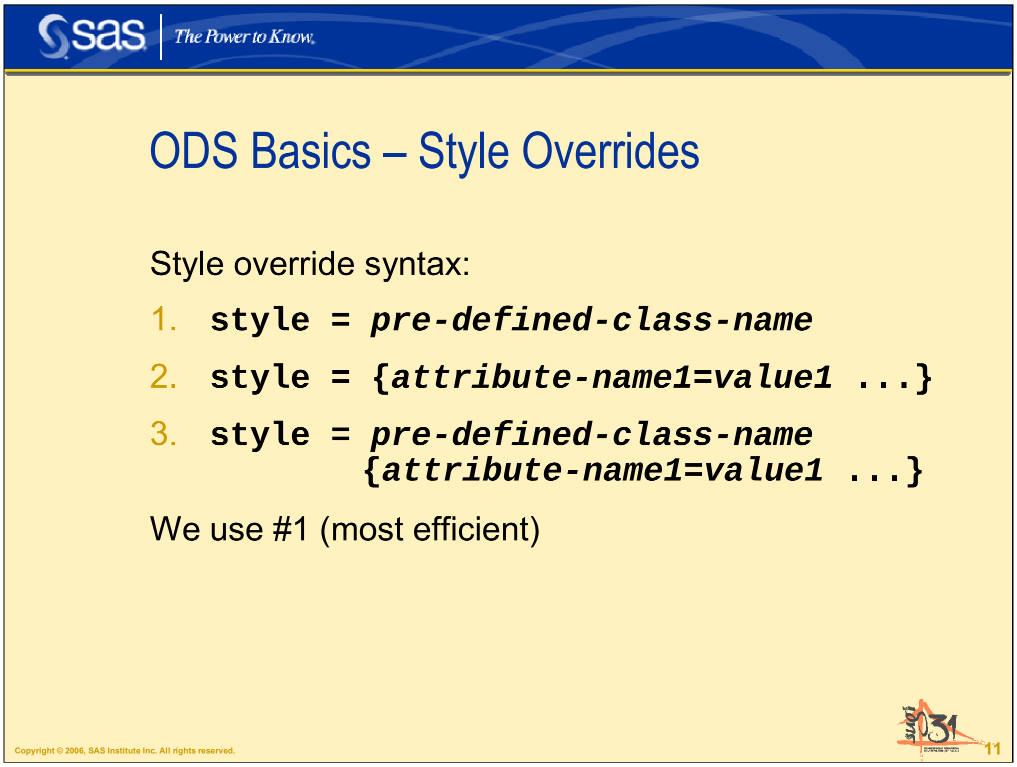
- Supported by PRINT, TABULATE and REPORT
- Change any ODS style attribute via STYLE=
- Example:
  `style = {font_style=italic background=blue}`
- Refer to the ODS documentation for a list of supported attributes
- Refer to PRINT, TABULATE and REPORT doc for sample usage

ODS style overrides are used to override attributes of the ODS style you are using. For example, you could use a style override to change the fonts or colors used by the SUGI31 style. But it is far more efficient do that with a style.

ODS Documentation:

"Chapter 5: The TEMPLATE Procedure", *The Complete Guide to the SAS Output Delivery System, Version 8* (see "The STYLE Statement")

"Chapter 9: TEMPLATE Procedure: Creating a Style Definition", *SAS 9.1 Output Delivery System, User's Guide*
(www.sas.com/apps/pubscat/bookdetails.jsp?catid=1&pc=58966)



ODS Basics – Style Overrides

Style override syntax:

1. `style = pre-defined-class-name`
2. `style = {attribute-name1=value1 ...}`
3. `style = pre-defined-class-name`
   `{attribute-name1=value1 ...}`

We use #1 (most efficient)

Above you see the syntax for using style overrides.

The second form is what you might use to change the fonts or colors used by a style class, as illustrated on the previous slide. While this form is the one that is most commonly used, it is also the least efficient. That is because, if overused, it can cause ODS to create files that are much larger than they would be if the style override was not used.

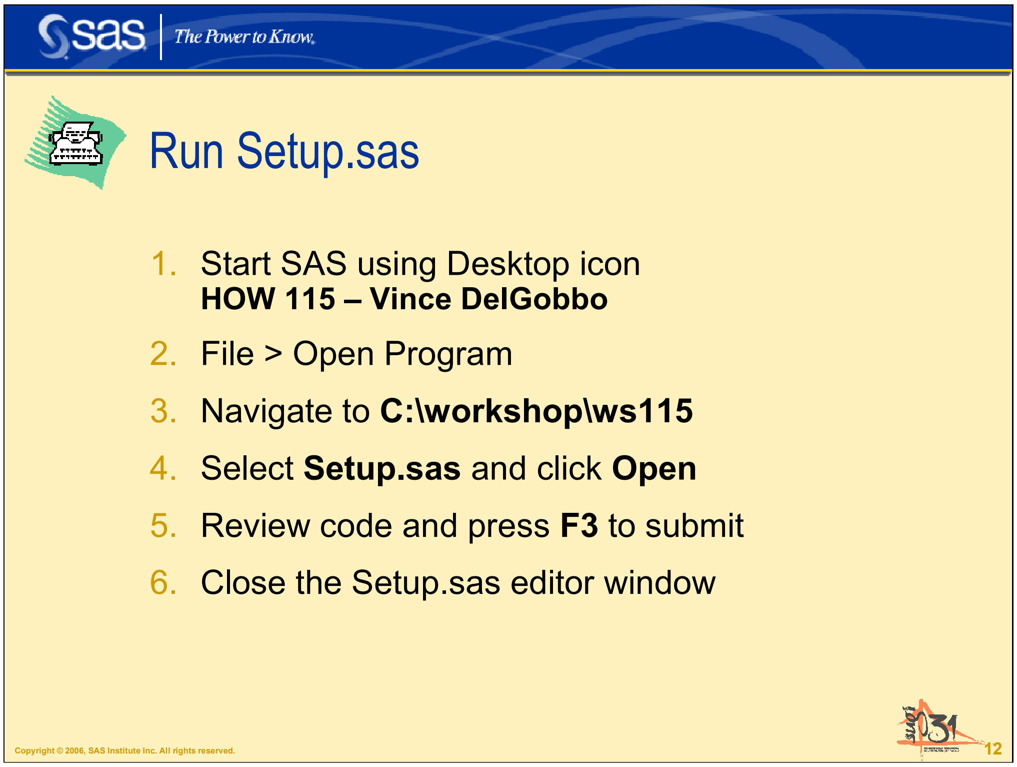
The first form is the most efficient, and the one that we will use in this workshop. It is used to associate a pre-defined ODS style class definition with a particular piece of output.

The SUGI31 style defined in the file "Setup.sas" contains a number of different style classes, such as "header_box" and "data_z8". We will associate these style classes with discrete parts of the SAS output using syntax such as:

```
style = header_box
```

and

```
style = data_z8
```



sas The Power to Know.

Run Setup.sas

1. Start SAS using Desktop icon
   HOW 115 – Vince DelGobbo
2. File > Open Program
3. Navigate to **C:\workshop\ws115**
4. Select **Setup.sas** and click **Open**
5. Review code and press **F3** to submit
6. Close the Setup.sas editor window

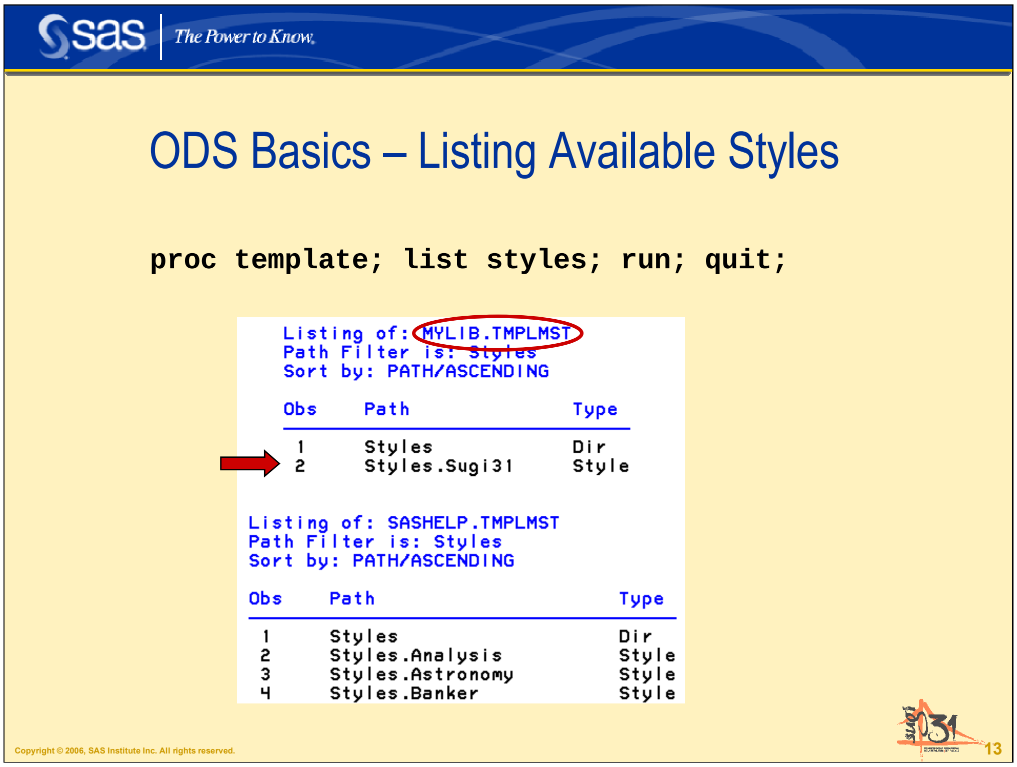
The SAMPDIR global macro variable specifies the directory containing our sample code and data, as well as the ODS-generated XML output

The program assigns a SAS library (SAMPLE) for the input data and one (MYLIB) for the output ODS tagsets and styles that we will be creating. While it is OK to use the WORK or SASUSER libraries to temporarily store custom 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.

The ODS PATH statement controls the search and storage locations for styles and tagsets. Thus, MYLIB will be searched/used first. The ODS tagset "ExcelXP" has undergone some important changes since SAS 9.1 was released, so we will import a new copy and store it in the MYLIB library.

Next, we load the SAS macro (XLXP2SAS) that imports Excel workbooks into SAS tables. You may wish to place this macro in your autocall library.

Finally, the SUGI31 style, which uses a color scheme similar to the SUGI31 conference materials and Web site, is created. Note the definition of custom style classes, such as "header_box" and "data_z8".



ODS Basics – Listing Available Styles

```
proc template; list styles; run; quit;
```

```
   Listing of: MYLIB.TMPLMST
   Path Filter is: Styles
   Sort by: PATH/ASCENDING

   Obs     Path                      Type
   ------------------------------------------
     1     Styles                    Dir
     2     Styles.Sugi31             Style

Listing of: SASHELP.TMPLMST
Path Filter is: Styles
Sort by: PATH/ASCENDING

Obs     Path                             Type
----------------------------------------------
  1     Styles                           Dir
  2     Styles.Analysis                  Style
  3     Styles.Astronomy                 Style
  4     Styles.Banker                    Style
```

This code is used to list the ODS styles available on your system. The image above shows a partial listing of the available styles.

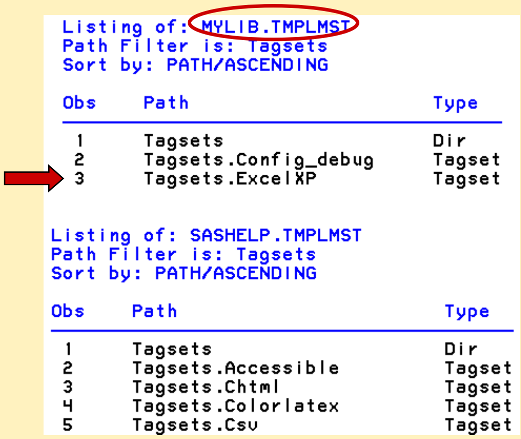
We will be using the user-defined style named "SUGI31".

Note that the MYLIB library appears first, due to the ODS PATH statement that was issued by the program "Setup.sas".

You can use the TEMPLATE procedure to create your own custom styles.

ODS Basics – Listing Available Tagsets

```
proc template; list tagsets; run; quit;
```



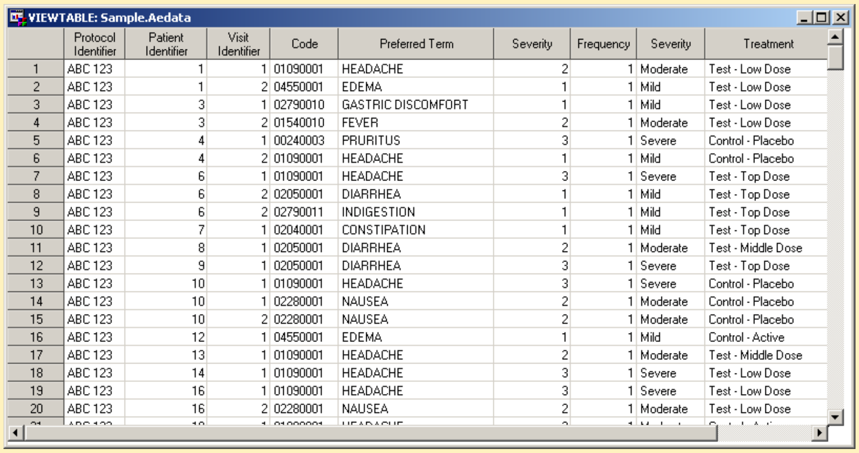
```
Listing of: MYLIB.TMPLMST
Path Filter is: Tagsets
Sort by: PATH/ASCENDING

Obs     Path                          Type
------------------------------------------
  1     Tagsets                       Dir
  2     Tagsets.Config_debug          Tagset
  3     Tagsets.ExcelXP               Tagset

Listing of: SASHELP.TMPLMST
Path Filter is: Tagsets
Sort by: PATH/ASCENDING

Obs     Path                          Type
------------------------------------------
 1      Tagsets                       Dir
 2      Tagsets.Accessible            Tagset
 3      Tagsets.Chtml                 Tagset
 4      Tagsets.Colorlatex            Tagset
 5      Tagsets.Csv                   Tagset
```

This code is used to list the ODS tagsets available on your system. The image above shows a partial listing of the available tagsets.

A copy of the ExcelXP tagset exists in both the MYLIB and SASHELP libraries. Since the MYLIB library appears first, the copy of the tagset located there will be used when referenced by your SAS code. This tagset appears first due to the ODS PATH statement that was issued by the program "Setup.sas".

You can use the TEMPLATE procedure to create your own custom tagsets.

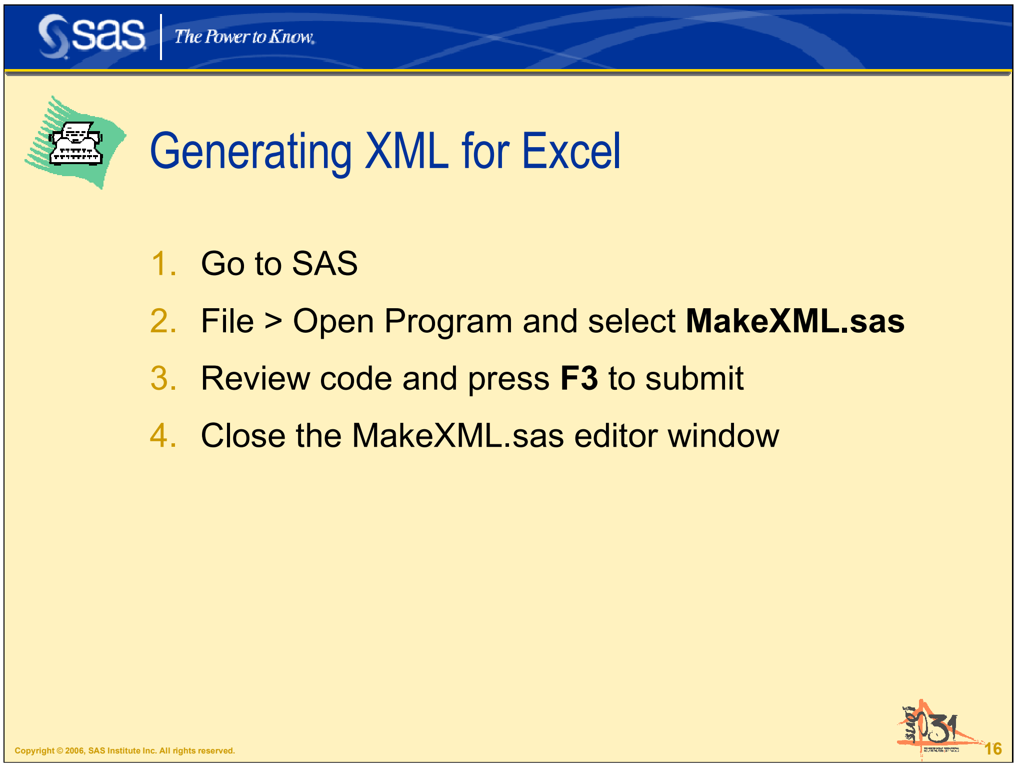
Preview Sample SAS Data



VIEWTABLE: Sample.Aedata

	Protocol Identifier	Patient Identifier	Visit Identifier	Code	Preferred Term	Severity	Frequency	Severity	Treatment
1	ABC 123	1	1	01090001	HEADACHE	2	1	Moderate	Test - Low Dose
2	ABC 123	1	2	04550001	EDEMA	1	1	Mild	Test - Low Dose
3	ABC 123	3	1	02790010	GASTRIC DISCOMFORT	1	1	Mild	Test - Low Dose
4	ABC 123	3	2	01540010	FEVER	2	1	Moderate	Test - Low Dose
5	ABC 123	4	1	00240003	PRURITUS	3	1	Severe	Control - Placebo
6	ABC 123	4	2	01090001	HEADACHE	1	1	Mild	Control - Placebo
7	ABC 123	6	1	01090001	HEADACHE	3	1	Severe	Test - Top Dose
8	ABC 123	6	2	02050001	DIARRHEA	1	1	Mild	Test - Top Dose
9	ABC 123	6	2	02790011	INDIGESTION	1	1	Mild	Test - Top Dose
10	ABC 123	7	1	02040001	CONSTIPATION	1	1	Mild	Test - Top Dose
11	ABC 123	8	1	02050001	DIARRHEA	2	1	Moderate	Test - Middle Dose
12	ABC 123	9	1	02050001	DIARRHEA	3	1	Severe	Test - Top Dose
13	ABC 123	10	1	01090001	HEADACHE	3	1	Severe	Control - Placebo
14	ABC 123	10	1	02280001	NAUSEA	2	1	Moderate	Control - Placebo
15	ABC 123	10	2	02280001	NAUSEA	2	1	Moderate	Control - Placebo
16	ABC 123	12	1	04550001	EDEMA	1	1	Mild	Control - Active
17	ABC 123	13	1	01090001	HEADACHE	2	1	Moderate	Test - Middle Dose
18	ABC 123	14	1	01090001	HEADACHE	3	1	Severe	Test - Low Dose
19	ABC 123	16	1	01090001	HEADACHE	3	1	Severe	Test - Low Dose
20	ABC 123	16	2	02280001	NAUSEA	2	1	Moderate	Test - Low Dose

This SAS table contains data pertaining to adverse events of a fictitious drug.



The Power to Know.

Generating XML for Excel

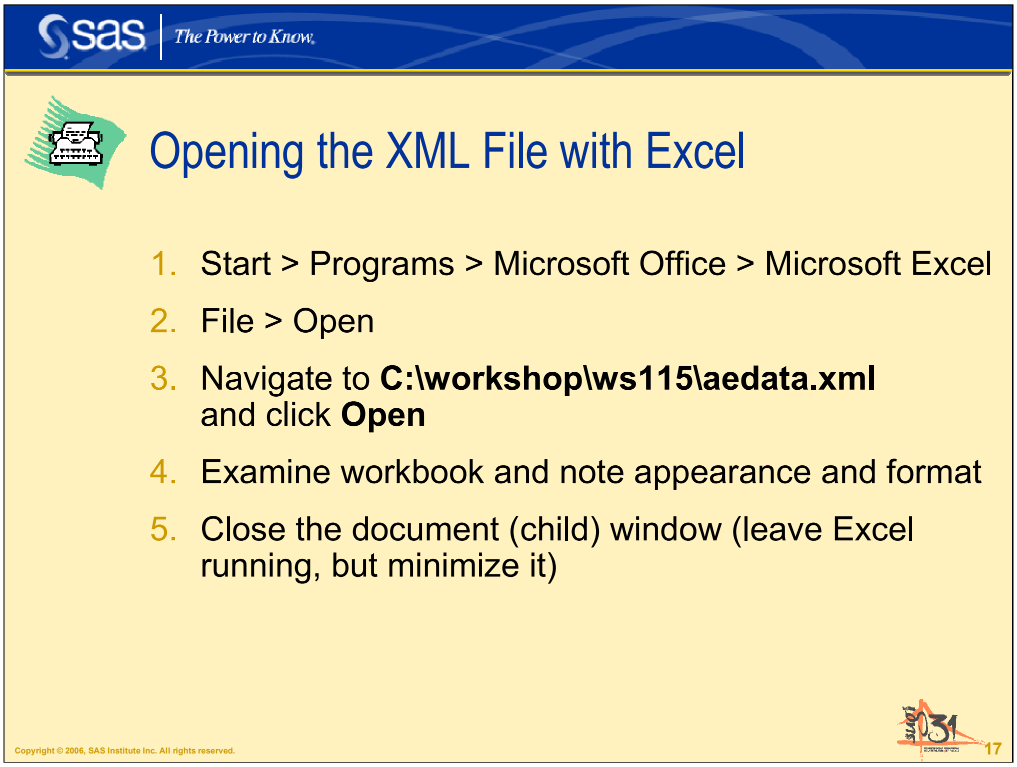
1. Go to SAS
2. File > Open Program and select **MakeXML.sas**
3. Review code and press **F3** to submit
4. Close the MakeXML.sas editor window

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 SUGI31 style controls the appearance of the output.

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

It may seem odd that the PRINT and TABULATE procedures are each run twice. If the WHERE clause is removed, then each could be run once, and the same output would be generated. As you will soon see, running the procedures multiple times provides the most flexibility when it comes to getting the exact type of output that we desire.

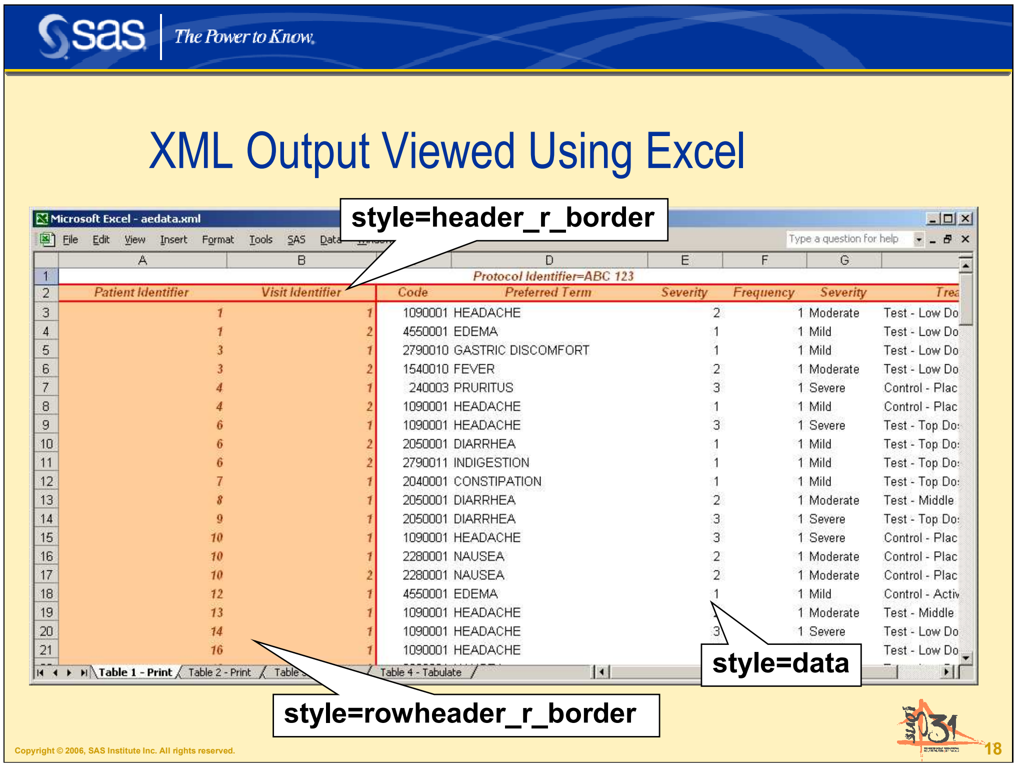


Opening the XML File with Excel

1. Start > Programs > Microsoft Office > Microsoft Excel
2. File > Open
3. Navigate to **C:\workshop\ws115\aedata.xml** and click **Open**
4. Examine workbook and note appearance and format
5. Close the document (child) window (leave Excel running, but minimize it)

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

When Excel reads the XML file, it renders the data as a multi-sheet workbook. Thus you can use all the editing and formatting features of Excel to modify the output and save it as a native Excel workbook
(File > Save As and choose **Microsoft Excel Workbook (*.xls)**)



XML Output Viewed Using Excel

style=header_r_border

style=data

style=rowheader_r_border

	A	B	C	D	E	F	G	
1				Protocol Identifier=ABC 123				
2	Patient Identifier	Visit Identifier	Code	Preferred Term	Severity	Frequency	Severity	Trea
3	1	1	1090001	HEADACHE	2	1	Moderate	Test - Low Do
4	1	2	4550001	EDEMA	1	1	Mild	Test - Low Do
5	3	1	2790010	GASTRIC DISCOMFORT	1	1	Mild	Test - Low Do
6	3	2	1540010	FEVER	2	1	Moderate	Test - Low Do
7	4	1	240003	PRURITUS	3	1	Severe	Control - Plac
8	4	2	1090001	HEADACHE	1	1	Mild	Control - Plac
9	6	1	1090001	HEADACHE	3	1	Severe	Test - Top Do:
10	6	2	2050001	DIARRHEA	1	1	Mild	Test - Top Do:
11	6	2	2790011	INDIGESTION	1	1	Mild	Test - Top Do:
12	7	1	2040001	CONSTIPATION	1	1	Mild	Test - Top Do:
13	8	1	2050001	DIARRHEA	2	1	Moderate	Test - Middle
14	9	1	2050001	DIARRHEA	3	1	Severe	Test - Top Do:
15	10	1	1090001	HEADACHE	3	1	Severe	Control - Plac
16	10	1	2280001	NAUSEA	2	1	Moderate	Control - Plac
17	10	2	2280001	NAUSEA	2	1	Moderate	Control - Plac
18	12	1	4550001	EDEMA	1	1	Mild	Control - Activ
19	13	1	1090001	HEADACHE		1	Moderate	Test - Middle
20	14	1	1090001	HEADACHE	3	1	Severe	Test - Low Do
21	16	1	1090001	HEADACHE				Test - Low Do

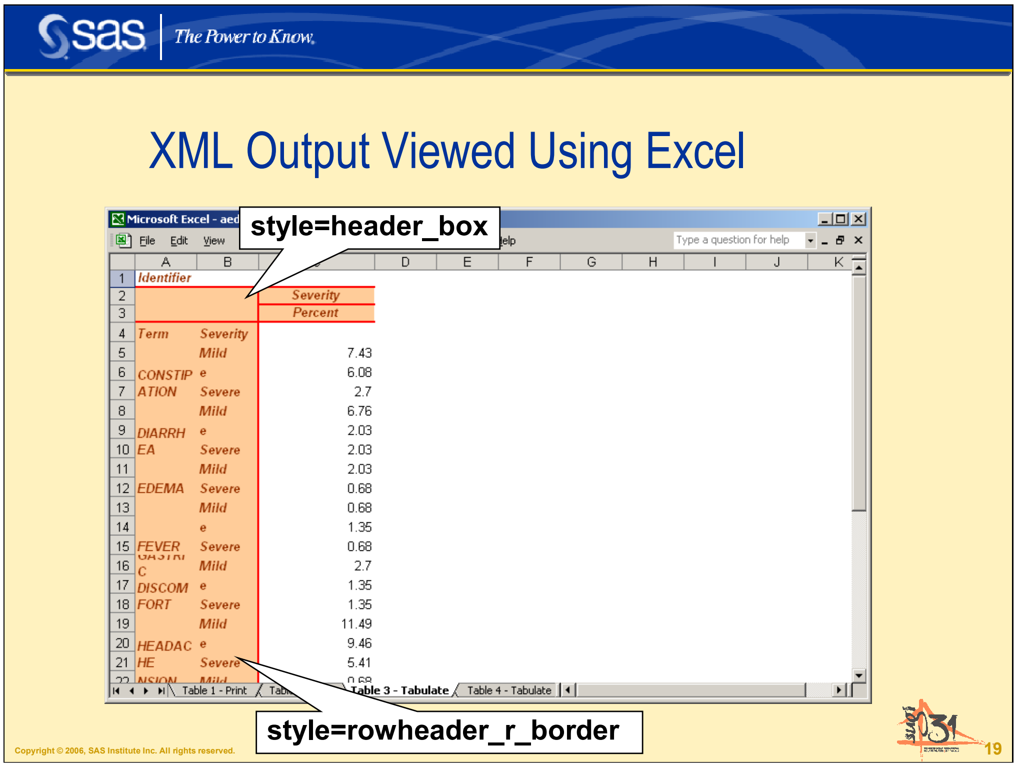
Problems with the output:

- The leading zeroes in the column labeled "Code" are missing.
- Default worksheet names were used.
- Some columns are wider than necessary, resulting in a very wide worksheet.
- The BY group label is centered, instead of left-justified.
- AutoFilters are missing from all columns.
- If you were to scroll downward in this worksheet, you will "loose" the column headers because by default, they are not "frozen".

ODS style classes:

The ODS style class used for the column heading of the "Visit Identifier" column is "header_r_border", while the "rowheader_r_border" class was used for the body of this column. These class names were used because of the style overrides applied in the PRINT procedure code.

All other class names used are the defaults chosen by ODS: "header" for the remaining column headers, "rowheader" for the row headers, and "data" for the body of the data section. Later, we will assign a different class to be used for the "Code" column.



XML Output Viewed Using Excel

style=header_box

	A	B	
1	Identifier		
2			Severity
3			Percent
4	Term	Severity	
5		Mild	7.43
6	CONSTIP	e	6.08
7	ATION	Severe	2.7
8		Mild	6.76
9	DIARRH	e	2.03
10	EA	Severe	2.03
11		Mild	2.03
12	EDEMA	Severe	0.68
13		Mild	0.68
14		e	1.35
15	FEVER	Severe	0.68
16	GASTRI C	Mild	2.7
17	DISCOM	e	1.35
18	FORT	Severe	1.35
19		Mild	11.49
20	HEADAC	e	9.46
21	HE	Severe	5.41

style=rowheader_r_border

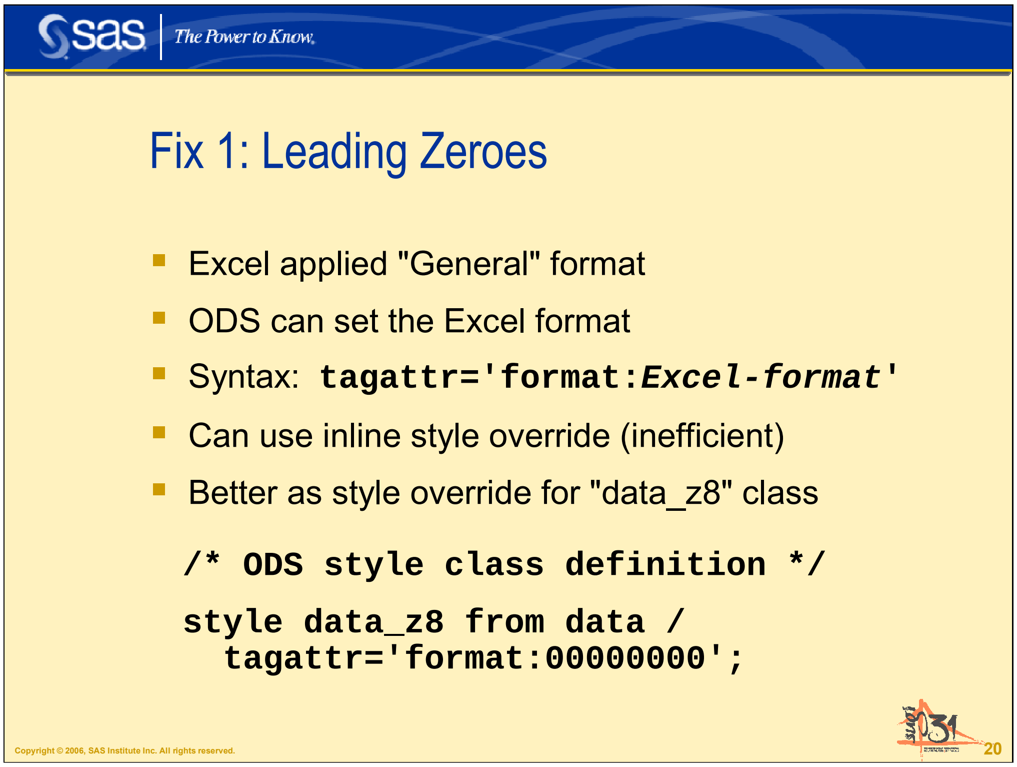
Problems with the output:

- Default worksheet names were used.
- Some columns are not as wide as necessary, resulting in the loss of data.
- The BY group label is truncated.

ODS style classes:

The ODS style class used for the "box cell" is "header_box", while the "rowheader_r_border" class was used for the body of the "Severity" column. These class names were used because of the style overrides applied in the TABULATE procedure code.

All other styles used are the defaults chosen by ODS: "header" for the column headers, "rowheader" for the remaining row headers, and "data" for the body of the data section.



Fix 1: Leading Zeroes

- Excel applied "General" format
- ODS can set the Excel format
- Syntax: `tagattr='format:Excel-format'`
- Can use inline style override (inefficient)
- Better as style override for "data_z8" class

```
/* ODS style class definition */

style data_z8 from data /
  tagattr='format:00000000';
```

The "Code" column in the PRINT procedure output contains an integer with leading zeroes. Upon opening the XML file, Excel applies the "General" format to this column, resulting in the loss of the leading zeroes.

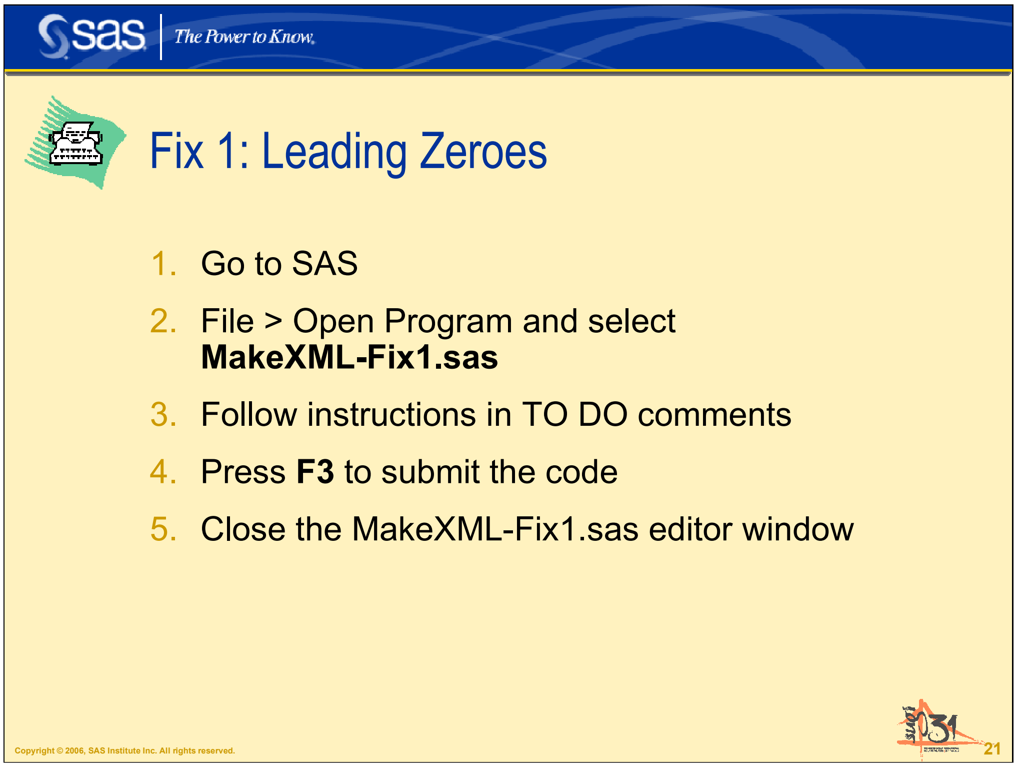
The ODS TAGATTR style attribute, in conjunction with the ExcelXP tagset, can be used to assign the Excel format that you want to use for this column.

You can specify this attribute as an inline style override, but as mentioned earlier, that method is inefficient. Instead, you would modify the ODS style and define a new class named "data_z8", as shown above. Then you would specify this style class name in your style override for the "Code" column.

Note that the SUGI31 style already contains the class named "data_z8", as it was defined in the style definition in the program "Setup.sas". So we can just use it.

You can find out more about Excel formats in the paper "A Beginner's Guide to Incorporating SAS® Output into Microsoft Office Applications"

(www2.sas.com/proceedings/sugi28/052-28.pdf)



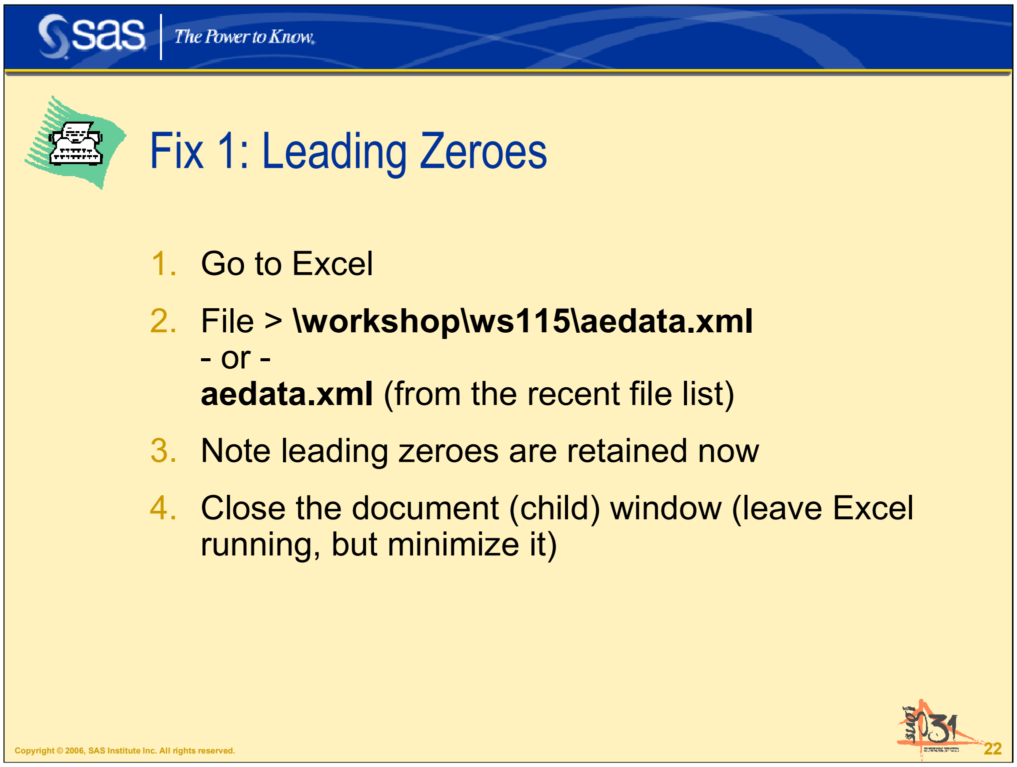
Fix 1: Leading Zeroes

1. Go to SAS
2. File > Open Program and select **MakeXML-Fix1.sas**
3. Follow instructions in TO DO comments
4. Press **F3** to submit the code
5. Close the MakeXML-Fix1.sas editor window

The only change we need to make is to apply the style override to the 2 VAR statements for the "aecode" column. The 2 VAR statements of PROC PRINT should look like this:

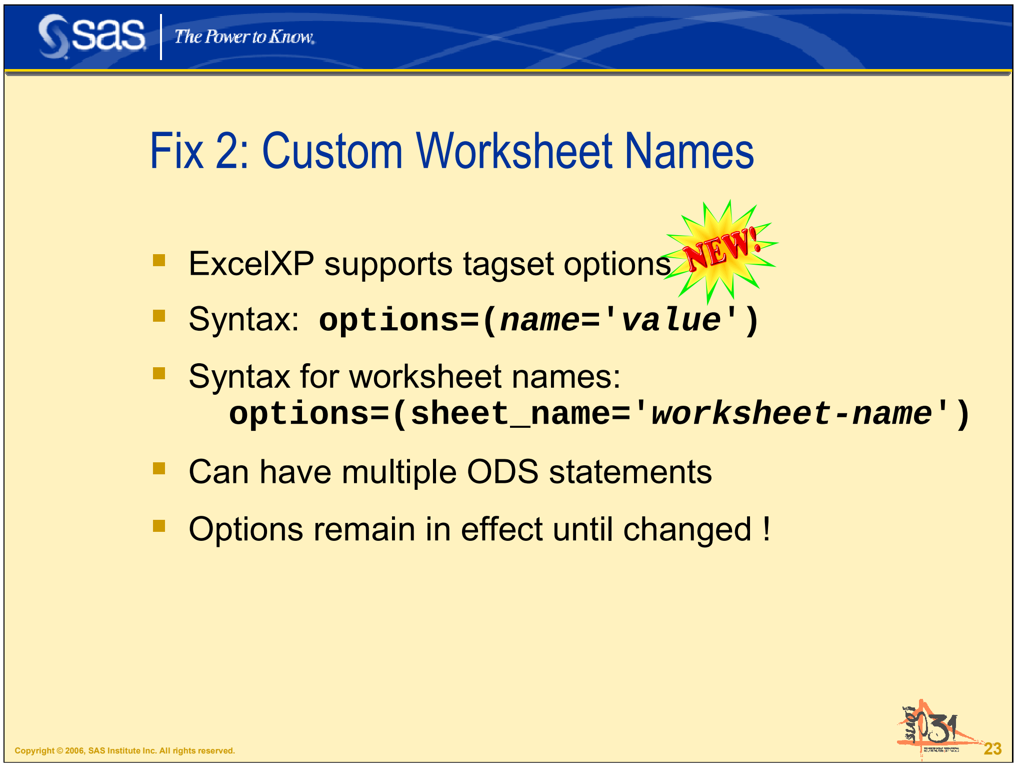
```
var aecode / style(data)=data_z8;
```

Note that we are specifying the name of a style class – data_z8 – and not a style attribute. This is the most efficient use of style overrides.



Fix 1: Leading Zeroes

1. Go to Excel
2. File > **\workshop\ws115\aedata.xml**
   - or -
   aedata.xml (from the recent file list)
3. Note leading zeroes are retained now
4. Close the document (child) window (leave Excel running, but minimize it)



Fix 2: Custom Worksheet Names

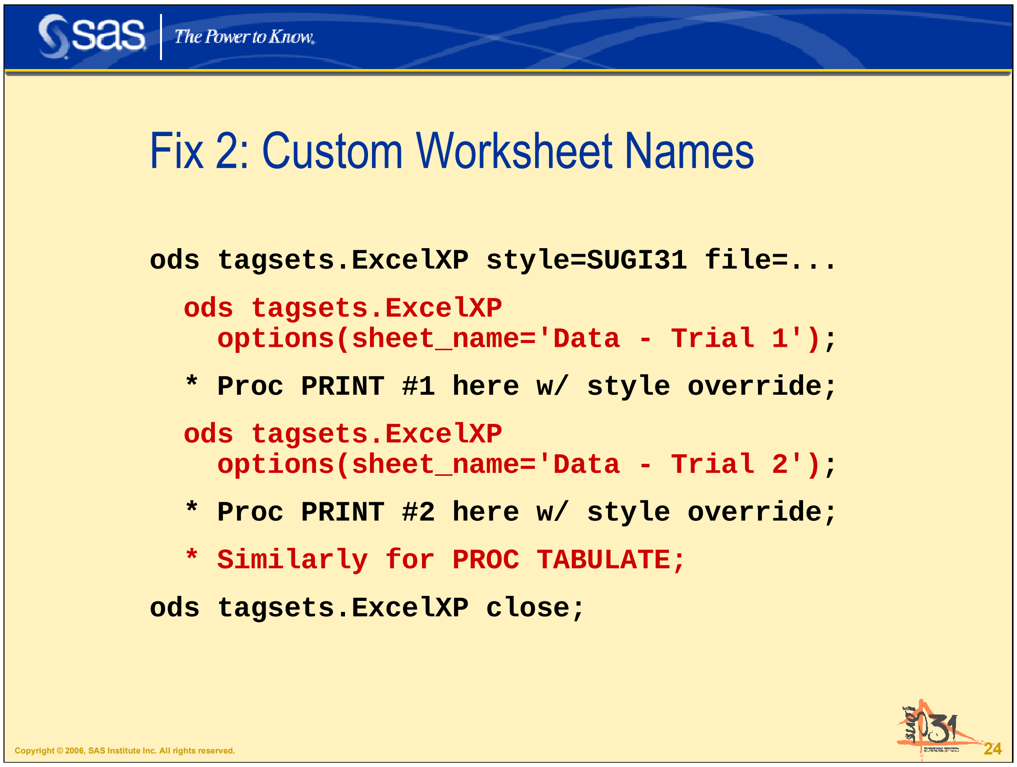
- ExcelXP supports tagset options
- Syntax: `options=(name='value')`
- Syntax for worksheet names:
  `options=(sheet_name='worksheet-name')`
- Can have multiple ODS statements
- Options remain in effect until changed !

The ExcelXP tagset supports a number of different options that control both the appearance and functionality of the Excel workbook. You simply add an OPTIONS attribute, shown above, to the ODS statement. Some of these options are not really new, but they have not been widely publicized or used.

The option to assign worksheet names is called SHEET_NAME. You specify the name you want to assign to the worksheet that is being created.

You can have more than one ODS statement. This is important, because it allows us to assign a different name for each of the 4 worksheets that are being created, as you will see on the next slide.

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



Fix 2: Custom Worksheet Names

```
ods tagsets.ExcelXP style=SUGI31 file=...
  ods tagsets.ExcelXP
    options(sheet_name='Data - Trial 1');
  * Proc PRINT #1 here w/ style override;
  ods tagsets.ExcelXP
    options(sheet_name='Data - Trial 2');
  * Proc PRINT #2 here w/ style override;
  * Similarly for PROC TABULATE;
ods tagsets.ExcelXP close;
```

This code snippet shows how you would specify a different name for each of the 4 worksheets in our sample workbook.

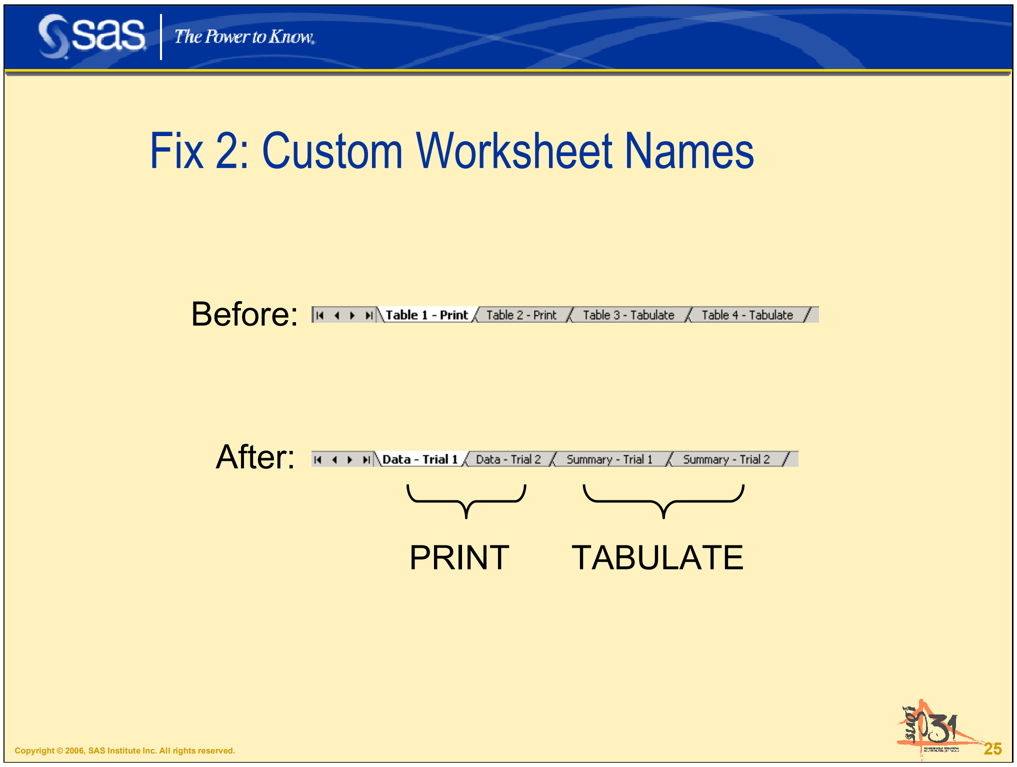
You would **add a new ODS statement** just before the first execution of the PRINT procedure to specify the worksheet name associated with that procedure output.

Since we want the second worksheet to have a different name (a requirement of Excel), another ODS statement is added just before the second PRINT procedure, to specify the name of the worksheet associated with that output.

You follow the same steps for the TABULATE output, specifying a unique worksheet name just before each instance of the procedure is run.

Now you can see why we run the PRINT and TABULATE procedures twice each.

The code which includes this fix can be found in the file "MakeXML-Fix2.sas". Note that the change to correct the missing leading zeroes in the PRINT procedure output is also included in this file.



Fix 2: Custom Worksheet Names

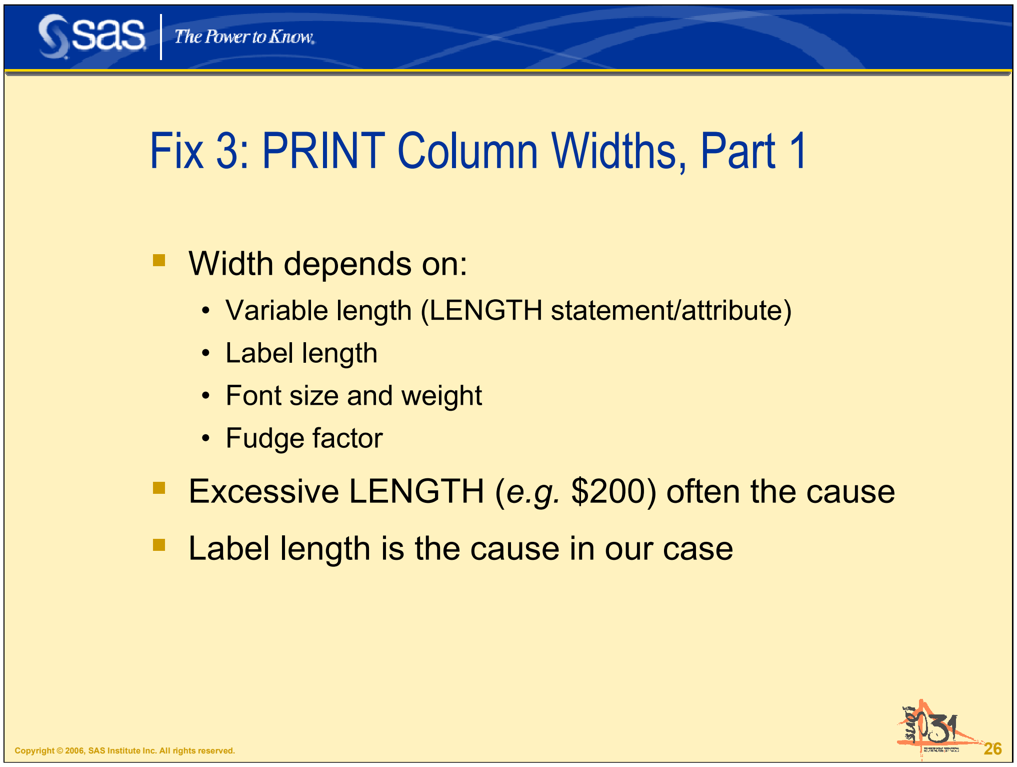
Before: Table 1 - Print | Table 2 - Print | Table 3 - Tabulate | Table 4 - Tabulate

After: Data - Trial 1 | Data - Trial 2 | Summary - Trial 1 | Summary - Trial 2

PRINT TABULATE

The images above show the worksheet names before and after the fix has been applied.

The worksheets contain the same procedure output as before, they just have different names.



sas The Power to Know.

Fix 3: PRINT Column Widths, Part 1

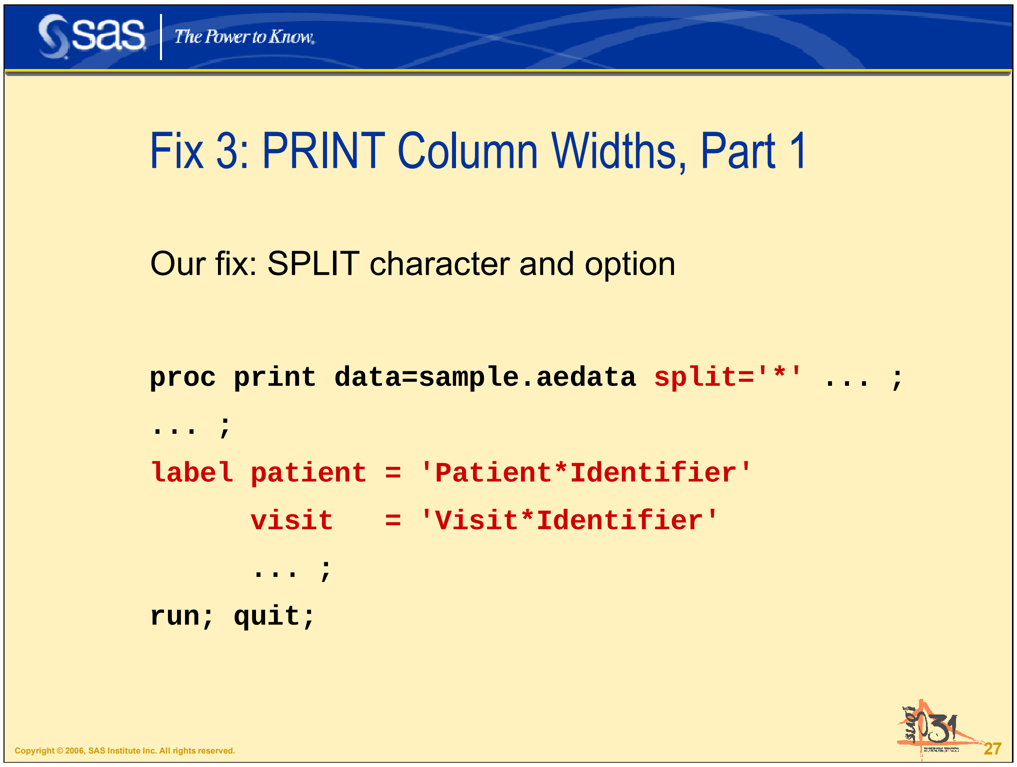
- Width depends on:
  - Variable length (LENGTH statement/attribute)
  - Label length
  - Font size and weight
  - Fudge factor
- Excessive LENGTH (*e.g.* $200) often the cause
- Label length is the cause in our case

The ExcelXP tagset estimates the column width based on several factors.

Often, a character column is defined in a SAS table with a LENGTH statement that is much larger than necessary. That is, the declared length is much larger than the actual text value. This is one common cause of Excel columns being too wide.

In our sample data, the defined column lengths are all reasonable, based on the data contained in the column. The cause for the excessive column width is the length of the column *labels*.

This can be easily remedied by assigning new column labels.



Fix 3: PRINT Column Widths, Part 1

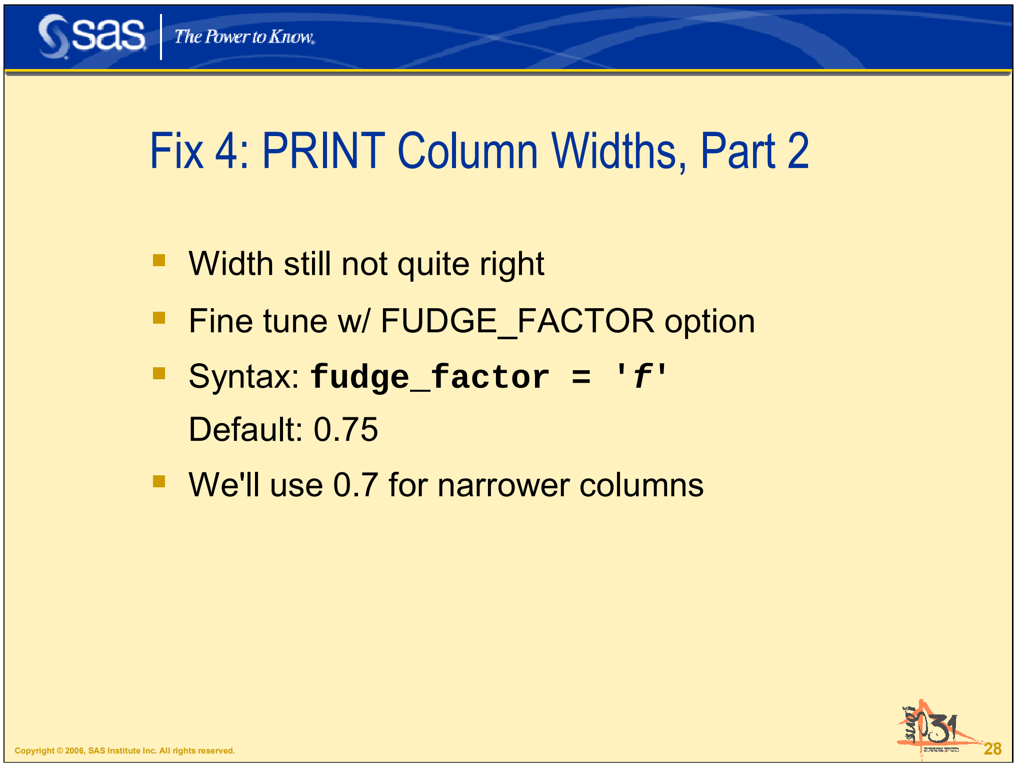
Our fix: SPLIT character and option

```
proc print data=sample.aedata split='*' ... ;
... ;
label patient = 'Patient*Identifier'
      visit   = 'Visit*Identifier'
      ... ;
run; quit;
```

By using a SPLIT character in your column label, you are effectively creating a smaller column.

In the case of the PATIENT column, the ExcelXP tagset now calculates the column width based on "Identifier", the largest single text string in the label. In our original PROC PRINT code, the tagset calculated the column width based on "Patient Identifier", which was the original label.

The code which includes this fix can be found in the file "MakeXML-Fix3.sas". Also note that all the changes made up to this point have been included in this file: the correction for leading zeroes in the PRINT procedure output and custom worksheet names.



Fix 4: PRINT Column Widths, Part 2

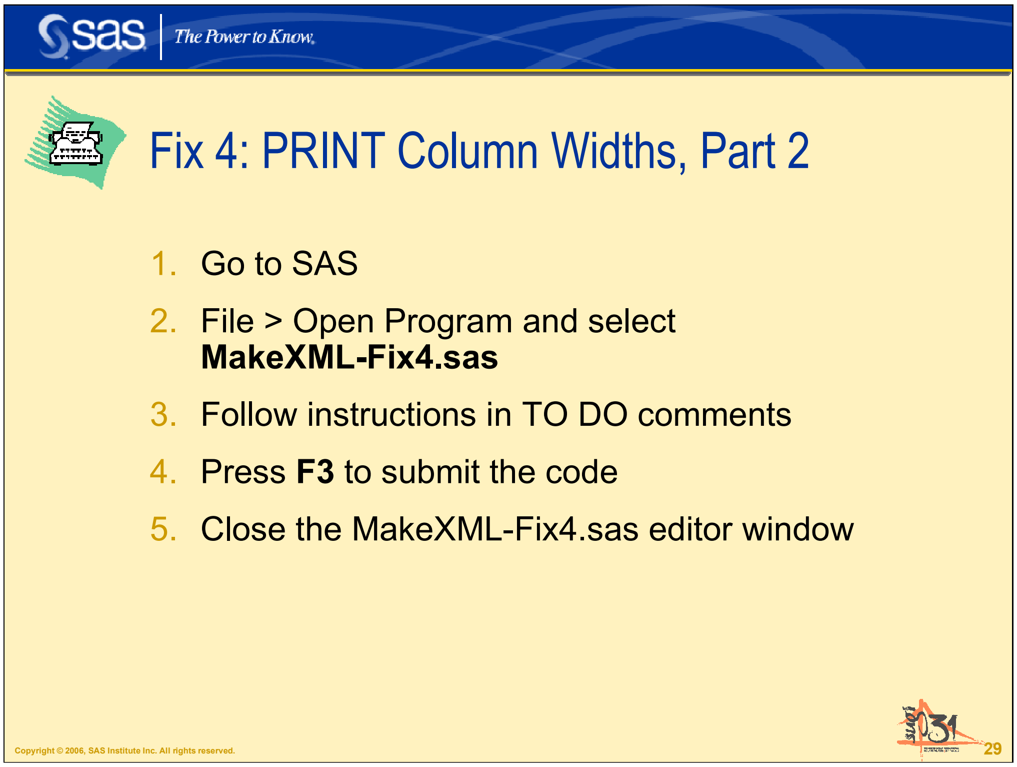
- Width still not quite right
- Fine tune w/ FUDGE_FACTOR option
- Syntax: **`fudge_factor = 'f'`**
  Default: 0.75
- We'll use 0.7 for narrower columns

If you were to run the previous code and open the output file with Excel, you would notice that the column widths were still a little wider than necessary.

You can use the FUDGE_FACTOR tagset option to fine tune the column widths. The default value for this tagset option is 0.75. Specifying a value less than 0.75 results in narrower columns, while a value of greater than 0.75 will create wider columns.

Recall that once tagset options are specified, they are in effect until they are changed.

We will specify `fudge_factor='0.7'` for the PRINT procedure output, but reset it to `fudge_factor='0.75'` for TABULATE procedure output.



Fix 4: PRINT Column Widths, Part 2

1. Go to SAS
2. File > Open Program and select **MakeXML-Fix4.sas**
3. Follow instructions in TO DO comments
4. Press **F3** to submit the code
5. Close the MakeXML-Fix4.sas editor window

The only change we need to make is to add the WIDTH_FUDGE option to the ODS statements preceding the PRINT and TABULATE procedure code.

The 2 ODS statements preceding PROC PRINT should have this option setting:

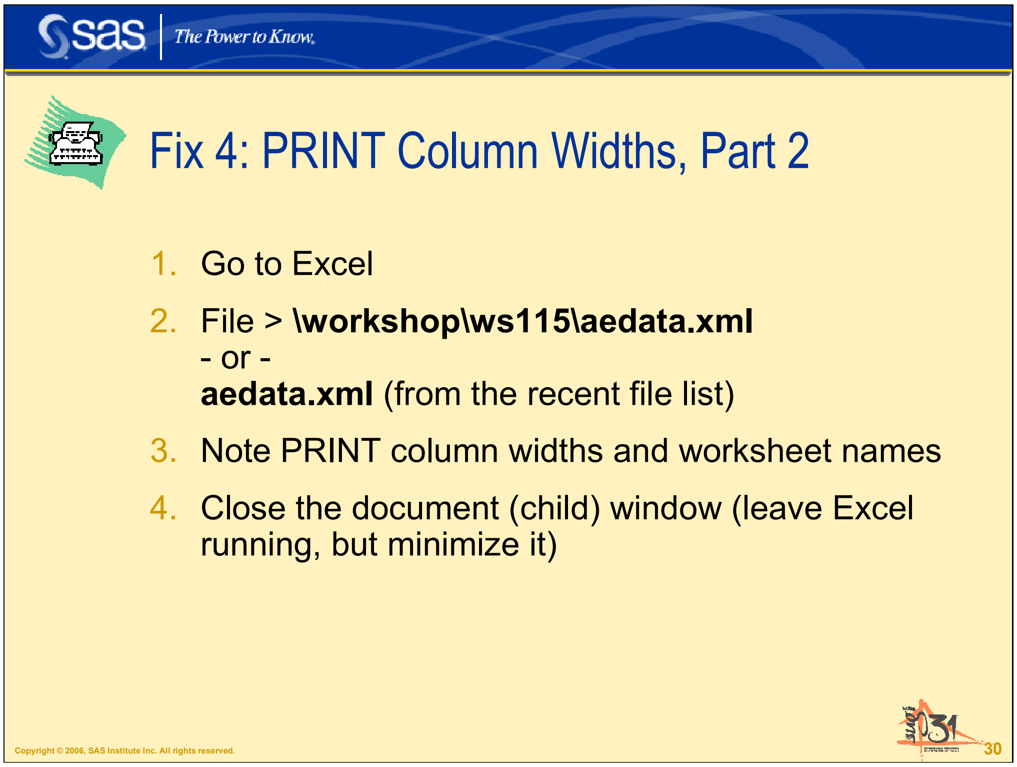
```
width_fudge='0.7'
```

The 2 ODS statements preceding PROC TABULATE should have this option setting :

```
width_fudge='0.75'
```

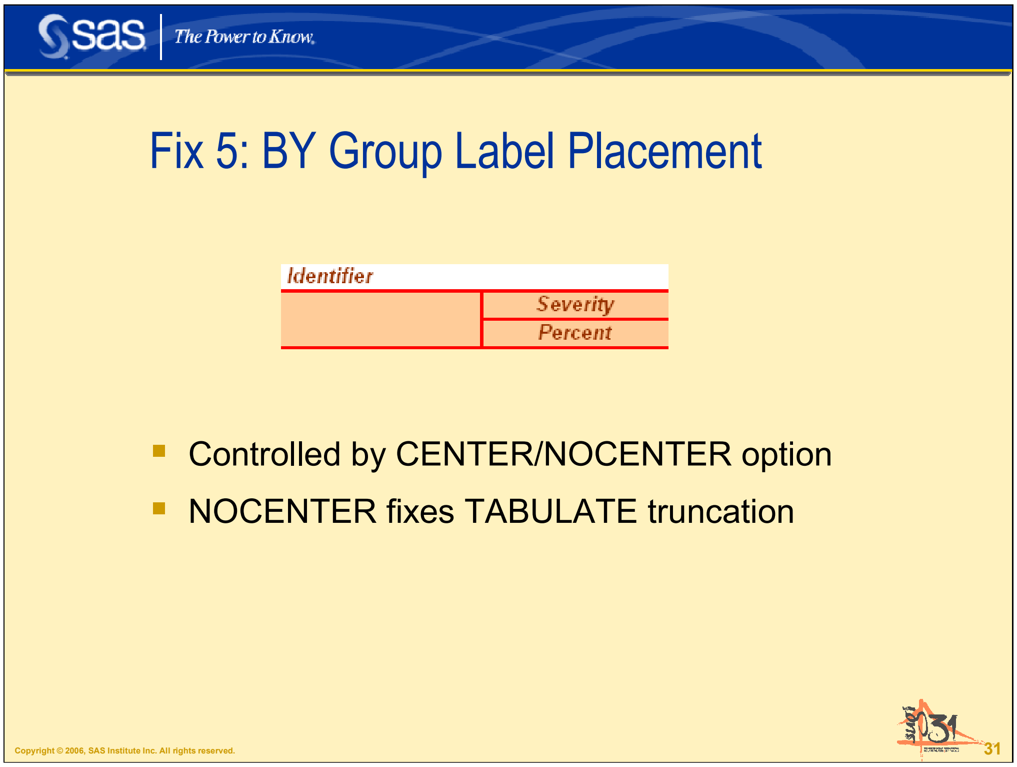
Note that the option name is already specified for you on the ODS statements – you just need to specify the appropriate option values.

Also note that all the changes made up to this point have been included in this file: the correction for leading zeroes in the PRINT procedure output, custom worksheet names and new column labels containing a SPLIT character.



Fix 4: PRINT Column Widths, Part 2

1. Go to Excel
2. File > **\workshop\ws115\aedata.xml**
   - or -
   aedata.xml (from the recent file list)
3. Note PRINT column widths and worksheet names
4. Close the document (child) window (leave Excel running, but minimize it)



The Power to Know.

Fix 5: BY Group Label Placement

Identifier

Severity

Percent

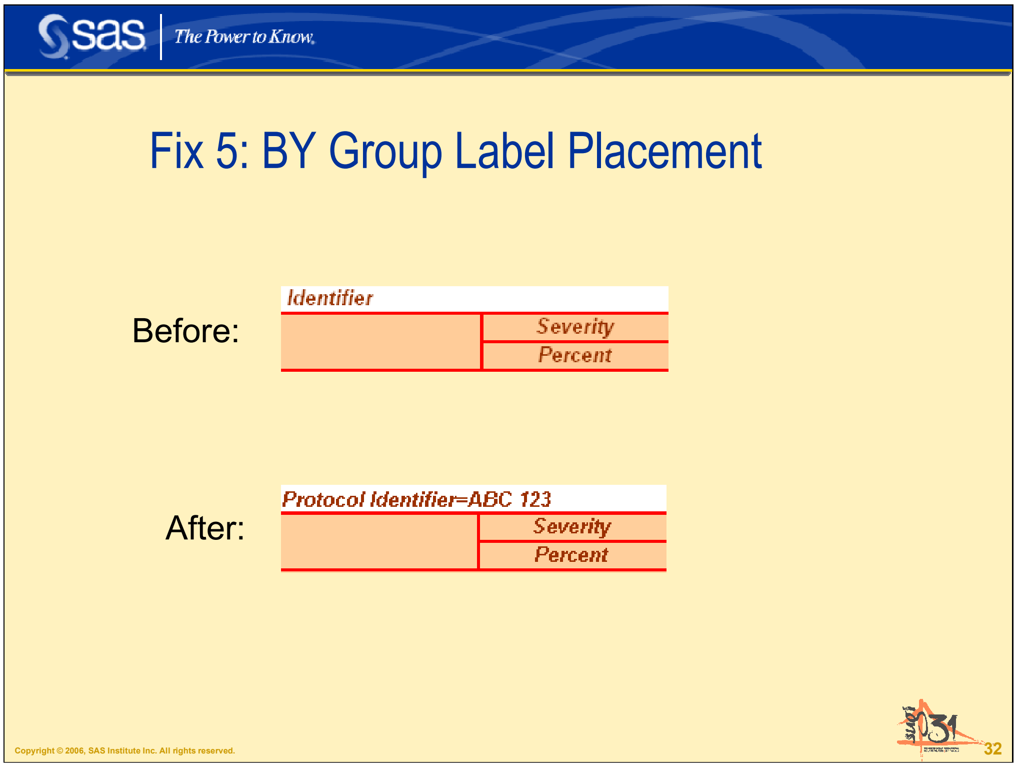
- Controlled by CENTER/NOCENTER option
- NOCENTER fixes TABULATE truncation

The image above shows what the BY group label looks like for the TABULATE procedure output. Note that the word "Protocol" is missing from the beginning of the label and the trial name is missing from the end. This data is actually in the document, but it is not visible.

The placement of the BY group label is controlled by the CENTER/NOCENTER SAS system option. In many installations, this value is set to CENTER, resulting in the placement shown above. This "truncation" is often observed when the CENTER option is active and label is long, but the SAS data in the table is narrow.

The easiest way to correct this problem is to specify the NOCENTER option prior to your ODS statements.

The code which includes this fix can be found in the file "MakeXML-Fix5.sas". Also note that all the changes made up to this point have been included in this file: the correction for leading zeroes in the PRINT procedure output, custom worksheet names and corrected column widths for PRINT procedure output.



The Power to Know.

Fix 5: BY Group Label Placement

Before:

Identifier

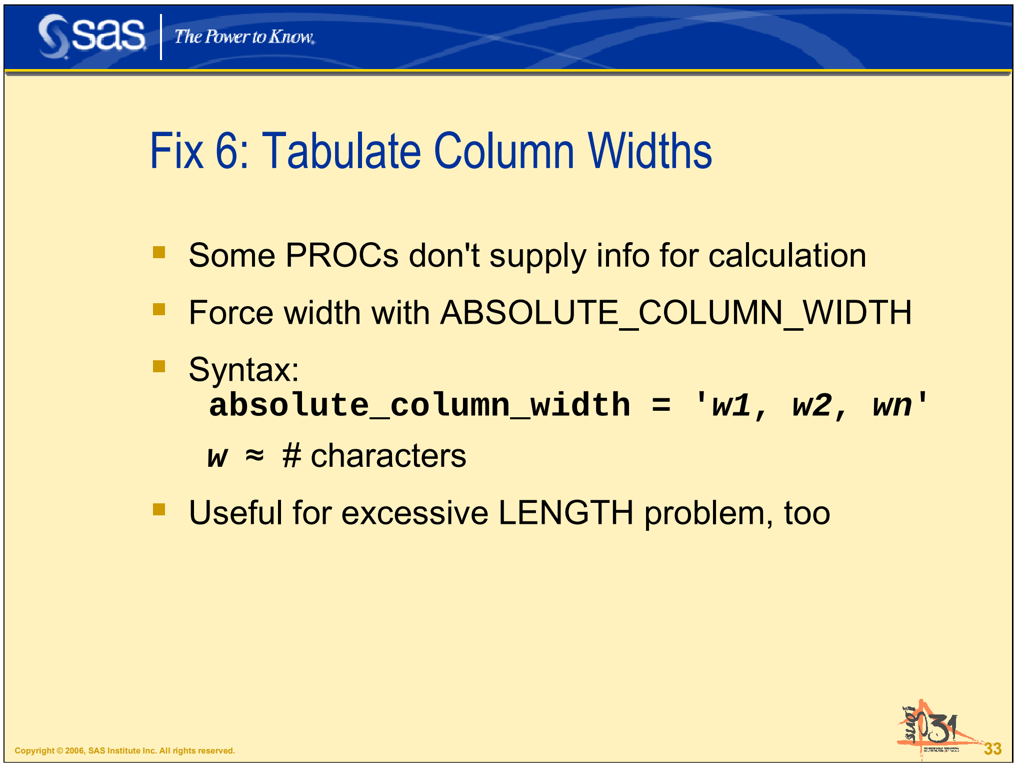
	Severity
	Percent

After:

Protocol Identifier=ABC 123

	Severity
	Percent

The images above show the BY group labels for the TABULATE procedure output before and after the NOCENTER option was specified.



Fix 6: Tabulate Column Widths

- Some PROCs don't supply info for calculation
- Force width with ABSOLUTE_COLUMN_WIDTH
- Syntax:
  `absolute_column_width = 'w1, w2, wn'`
  w ≈ # characters
- Useful for excessive LENGTH problem, too

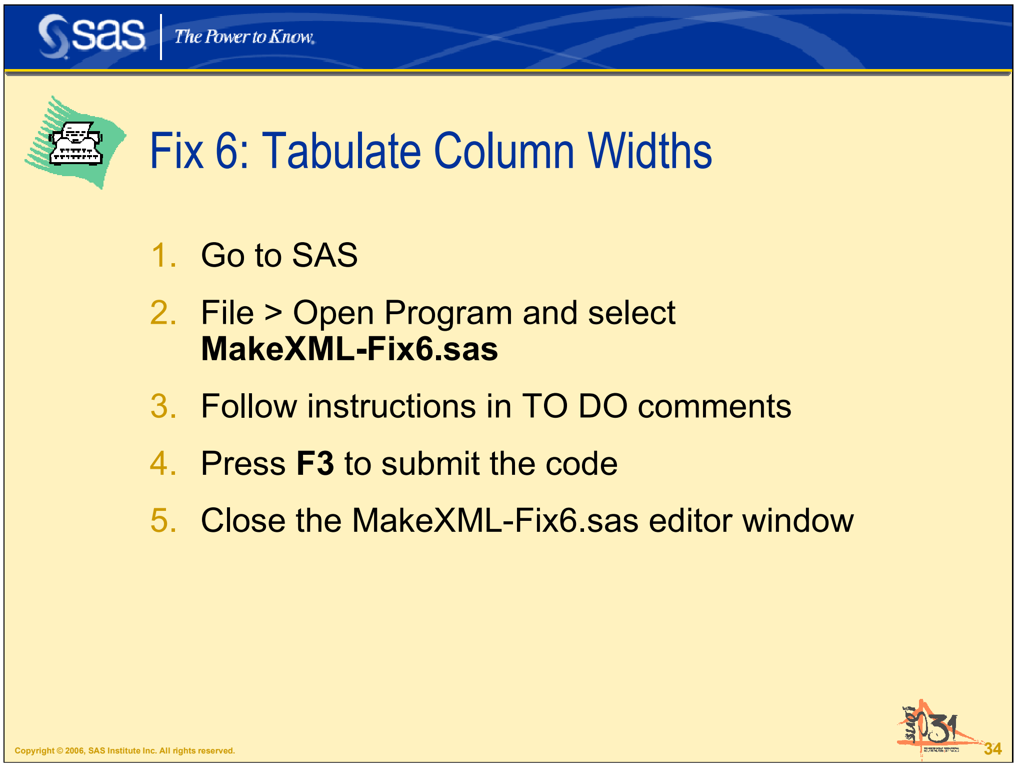
Some procedures, including TABULATE, do not supply enough information for the ExcelXP tagset to calculate a column width. It is expected that this will be changed in a future release of SAS.

In the mean time, you can use the ABSOLUTE_COLUMN_WIDTH option to specify the width of the columns. This option takes as input a comma-separated list of column widths, specified in the number of characters.

From examination of the TABULATE output, the maximum string length of the 3 columns are 21 ("ESOPHAGAEL IRRITATION"), 8 ("Moderate") and 8 ("Severity"). These are the values you would specify for the ABSOLUTE_COLUMN_WIDTH option.

There are instances where the values you calculate result in columns that are wider or narrower than necessary. In this case, make small adjustments to the values specified for ABSOLUTE_COLUMN_WIDTH until the widths are acceptable.

The ABSOLUTE_COLUMN_WIDTH option is also helpful for setting the column width when a SAS column was defined with an excessively large LENGTH statement (Example: $200 when the maximum string length is only 25).



The Power to Know.

Fix 6: Tabulate Column Widths

1. Go to SAS
2. File > Open Program and select **MakeXML-Fix6.sas**
3. Follow instructions in TO DO comments
4. Press **F3** to submit the code
5. Close the MakeXML-Fix6.sas editor window

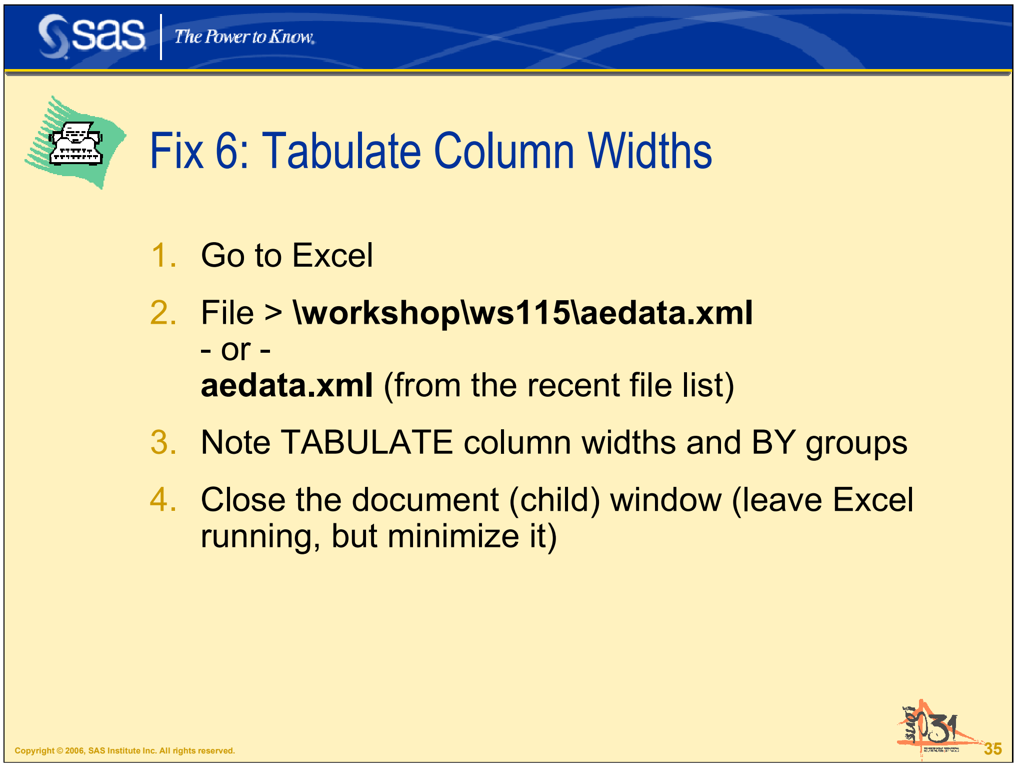
The only change we need to make is to add the ABSOLUTE_COLUMN_WIDTH option to the ODS statements preceding the TABULATE procedure code.

The 2 ODS statements preceding the PROC TABULATE code should have this option setting:

```
absolute_column_width='21, 8, 8'
```

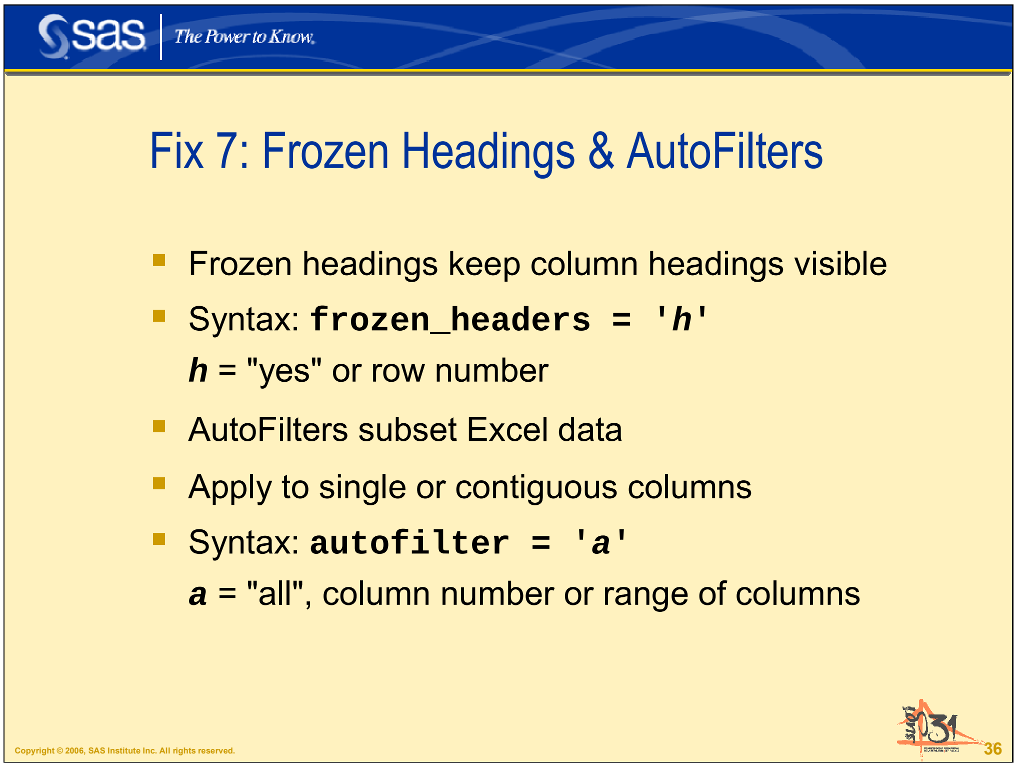
Note that the option name is already specified for you on the ODS statement – you just need to specify the appropriate option values.

Also note that all the changes made up to this point have been included in this file: the correction for leading zeroes, custom worksheet names, corrected column widths for PRINT procedure output, and corrected BY group labels.



Fix 6: Tabulate Column Widths

1. Go to Excel
2. File > **\workshop\ws115\aedata.xml**
   - or -
   aedata.xml (from the recent file list)
3. Note TABULATE column widths and BY groups
4. Close the document (child) window (leave Excel running, but minimize it)



Fix 7: Frozen Headings & AutoFilters

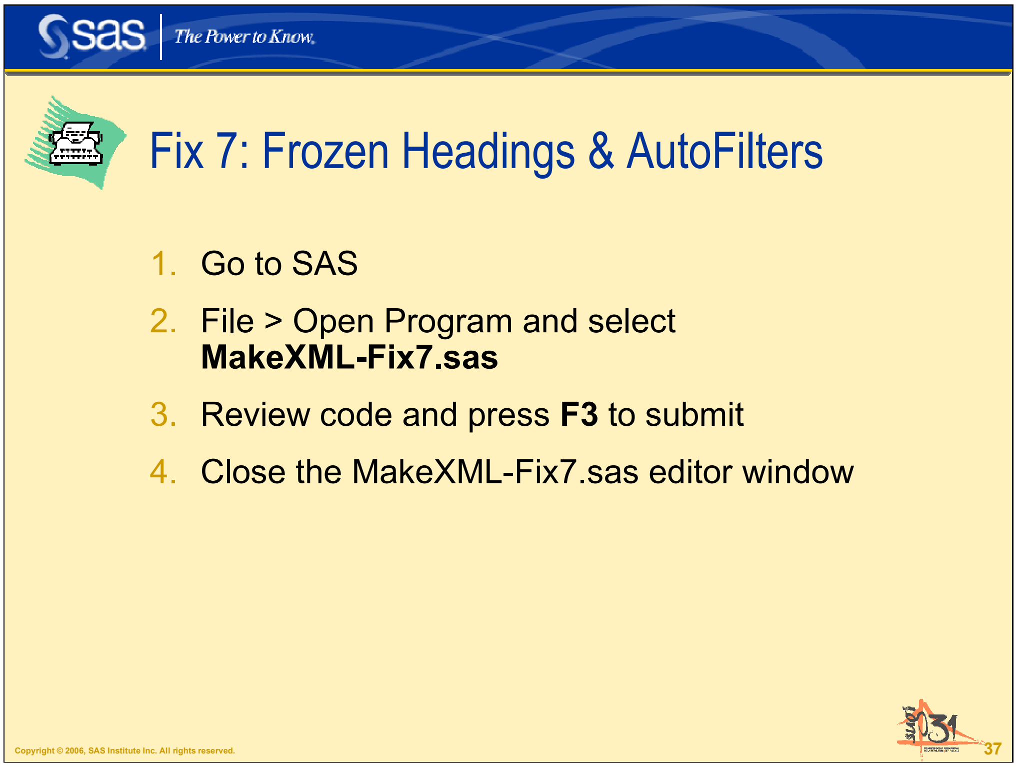
- Frozen headings keep column headings visible
- Syntax: **`frozen_headers = 'h'`**
  h = "yes" or row number
- AutoFilters subset Excel data
- Apply to single or contiguous columns
- Syntax: **`autofilter = 'a'`**
  a = "all", column number or range of columns

When you scroll down in a worksheet, the column headings can scroll off the top of the viewable screen. This is the case with our PRINT procedure output. To correct this problem, use the FROZEN_HEADERS option. This option specifies the rows that you want to "freeze" to serve as headings. If you specify "5", for example, rows 1-5 will be "frozen", and thus serve as the column headings when you scroll.

Excel has a wonderful feature known as an "AutoFilter". An AutoFilter allows you to filter, or subset data in a column. A column containing an AutoFilter is indicated by an arrow button in the column heading.

Suppose you want to view only records for patients who experienced a headache as an adverse event. You would apply an AutoFilter to the "Preferred Term" column by clicking the AutoFilter button and selecting **HEADACHE** from the list of values. Excel will automatically hide all rows that do not contain the text "HEADACHE" in the "Preferred Term" column. The data is still present in the worksheet, it is just hidden.

We will now apply frozen headers and AutoFilters to the PRINT procedure output, and experiment with them.



The Power to Know.

Fix 7: Frozen Headings & AutoFilters

1. Go to SAS
2. File > Open Program and select **MakeXML-Fix7.sas**
3. Review code and press **F3** to submit
4. Close the MakeXML-Fix7.sas editor window

You do not need to make any changes to the code – they have been made for you.

In reviewing the code, you will notice that the following options were added to the 2 ODS statements that precede the PRINT procedure code:

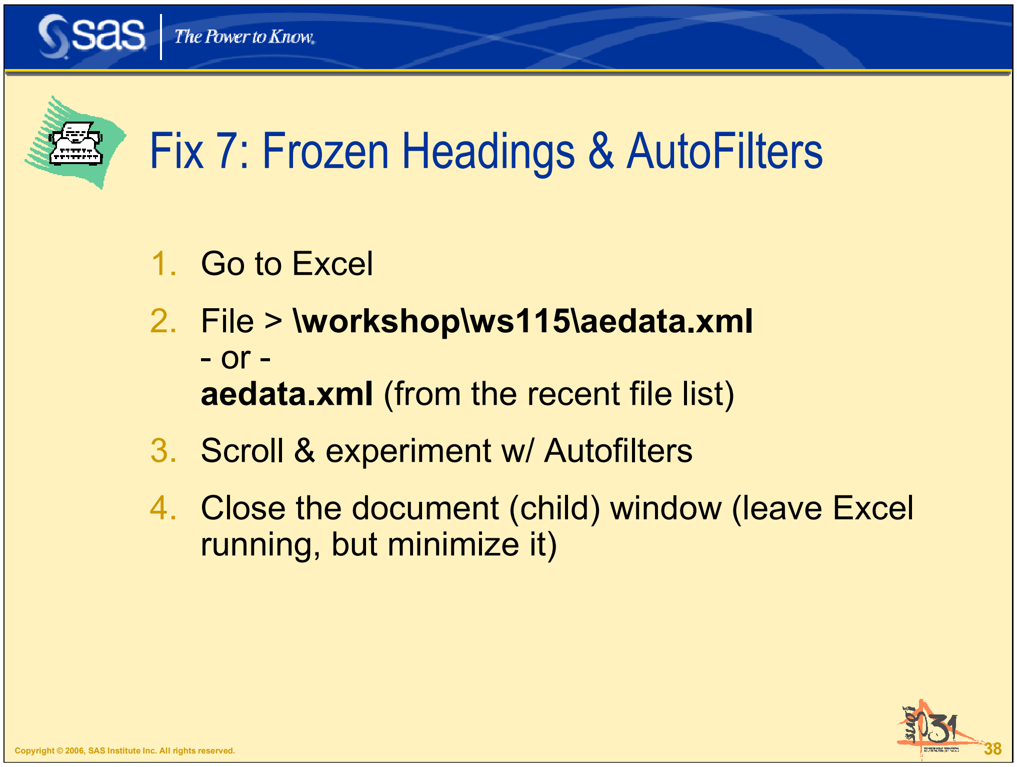
```
frozen_headers='2' autofilter='all'
```

This will cause rows 1-2 to act as column headings when you scroll through the worksheet, and AutoFilters will be applied to all columns.

Recall that tagset options remain in effect until they are changed. Since we don't want either of these options to affect the TABULATE output, the default option values have been added to the 2 ODS statements that precede the TABULATE procedure code:

```
frozen_headers='no' autofilter='none'
```

Finally, note that all the changes made up to this point have been included in this file: the correction for leading zeroes, custom worksheet names, corrected column widths for PRINT procedure output, corrected BY group labels, and corrected column widths in the TABULATE procedure output.



Fix 7: Frozen Headings & AutoFilters

1. Go to Excel
2. File > **\workshop\ws115\aedata.xml**
   - or -
   aedata.xml (from the recent file list)
3. Scroll & experiment w/ Autofilters
4. Close the document (child) window (leave Excel running, but minimize it)

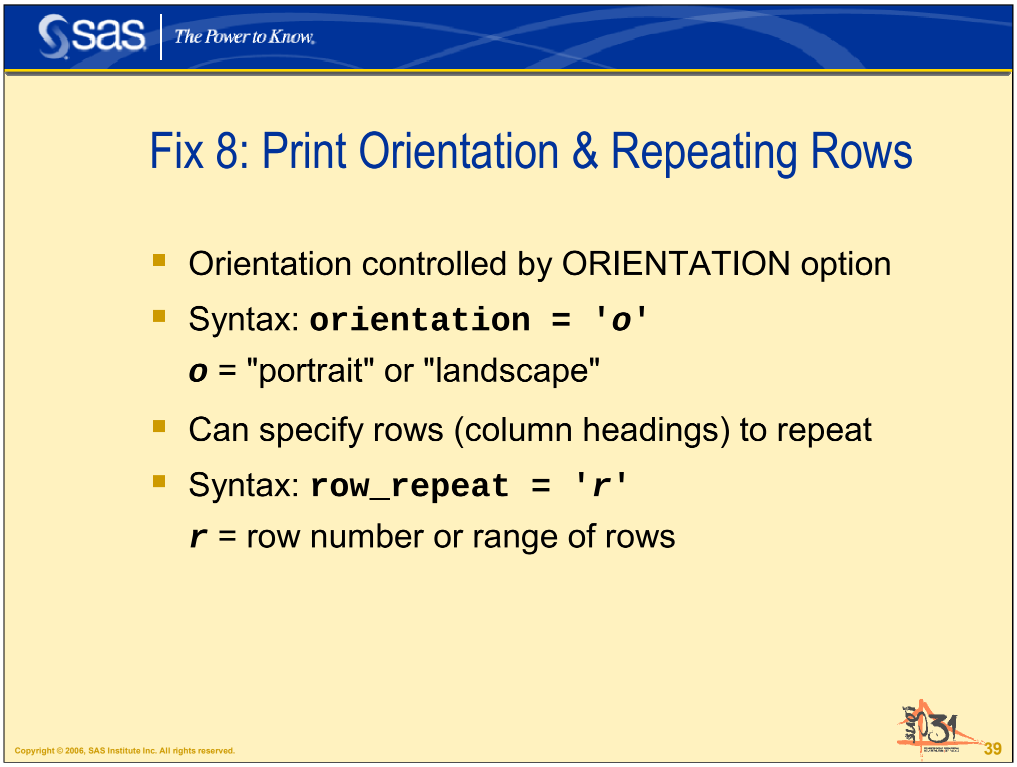
As you scroll up and down in the PRINT procedure output, notice that the column headings remain in place.

Scroll to the top of the worksheet and click the AutoFilter button in the "Preferred Term" column heading. Select **HEADACHE** from the list of values. Excel hides all rows that do not have the text "HEADACHE" in the "Preferred Term" column.

Next, click the AutoFilter button in the "Severity" column that contains the character values "Mild", "Moderate" and "Severe". Choose **Severe** from the list of values. Since AutoFilters are additive, now only data for patients who experienced severe headaches as an adverse event are displayed.

An AutoFilter is active when the color of the arrow in the AutoFilter button is blue, instead of black.

To clear an individual AutoFilter, click the AutoFilter button and select **(All)**. For example, to display data for patients who experienced *any* type of severe adverse event, click the AutoFilter button for the "Preferred Term" column and select **(All)**.



The Power to Know.

Fix 8: Print Orientation & Repeating Rows

- Orientation controlled by ORIENTATION option
- Syntax: **`orientation = 'o'`**

  o = "portrait" or "landscape"
- Can specify rows (column headings) to repeat
- Syntax: **`row_repeat = 'r'`**

  r = row number or range of rows

With all of the changes that have been put into place thus far, the Excel workbook should match the images shown at the beginning of this presentation. We will now look at some options that improve the appearance of *printed* output.

The PRINT procedure output is wide, and should be printed in landscape orientation, while the TABULATE procedure output will look fine if printed in portrait. You can set the print orientation using the ORIENTATION tagset option.

Just as frozen column headings were helpful when scrolling through the PRINT procedure output, displaying headings on all pages of printed output is also desirable. You can accomplish this by using the ROW_REPEAT option. For example, if you want rows 1 and 2 to repeat at the top of each printed page, specify `1-2` for this option.

The code which includes these changes can be found in the file "MakeXML-Fix8.sas". The following options were added to the 2 ODS statements that precede the PRINT procedure code:

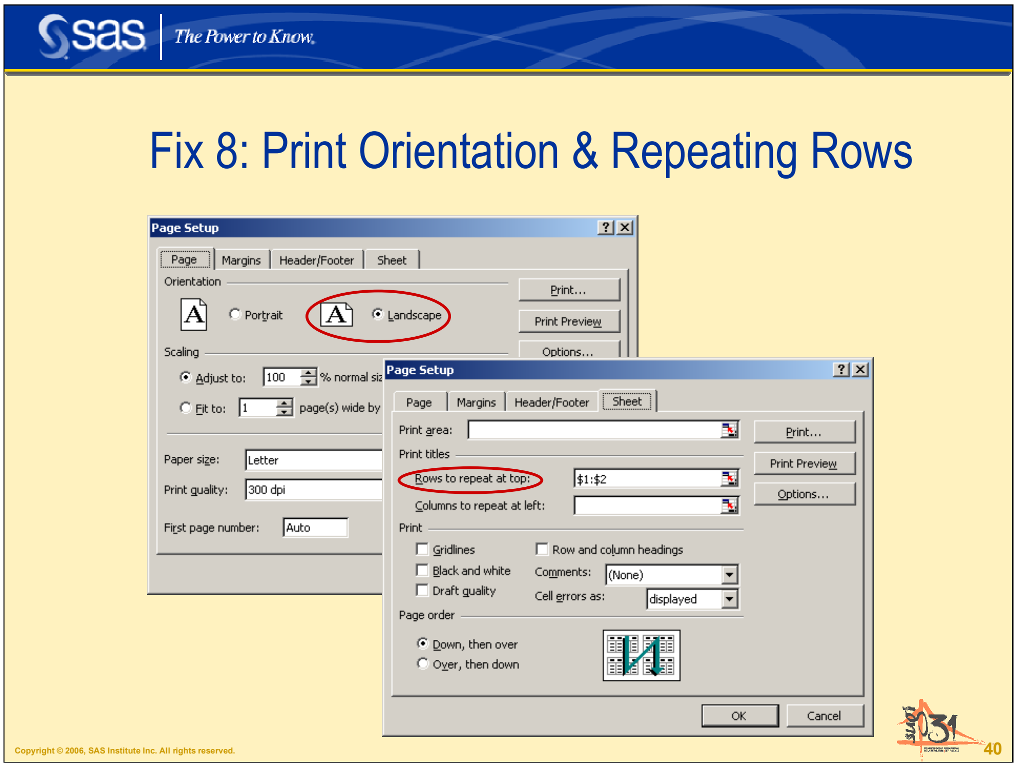
```
orientation='landscape' row_repeat='1-2'
```

Since we don't want different behavior for the TABULATE output, these options have been added to the 2 ODS statements that precede the TABULATE procedure code:

```
orientation='portrait' row_repeat='none'
```

Also note that all the changes made up to this point have been included in this file: the correction for leading zeroes, custom worksheet names, corrected column widths for PRINT procedure output, corrected BY group labels, corrected column widths in the TABULATE procedure output, frozen column headings and AutoFilters for the PRINT procedure output.

Fix 8: Print Orientation & Repeating Rows



Orientation

○ Portrait ● Landscape

Scaling

● Adjust to: 100 % normal siz

○ Fit to: 1 page(s) wide by

Paper size: Letter

Print quality: 300 dpi

First page number: Auto

Print area:

Print titles

Rows to repeat at top: $1:$2

Columns to repeat at left:

Print

☐ Gridlines ☐ Row and column headings

☐ Black and white Comments: (None)

☐ Draft quality Cell errors as: displayed

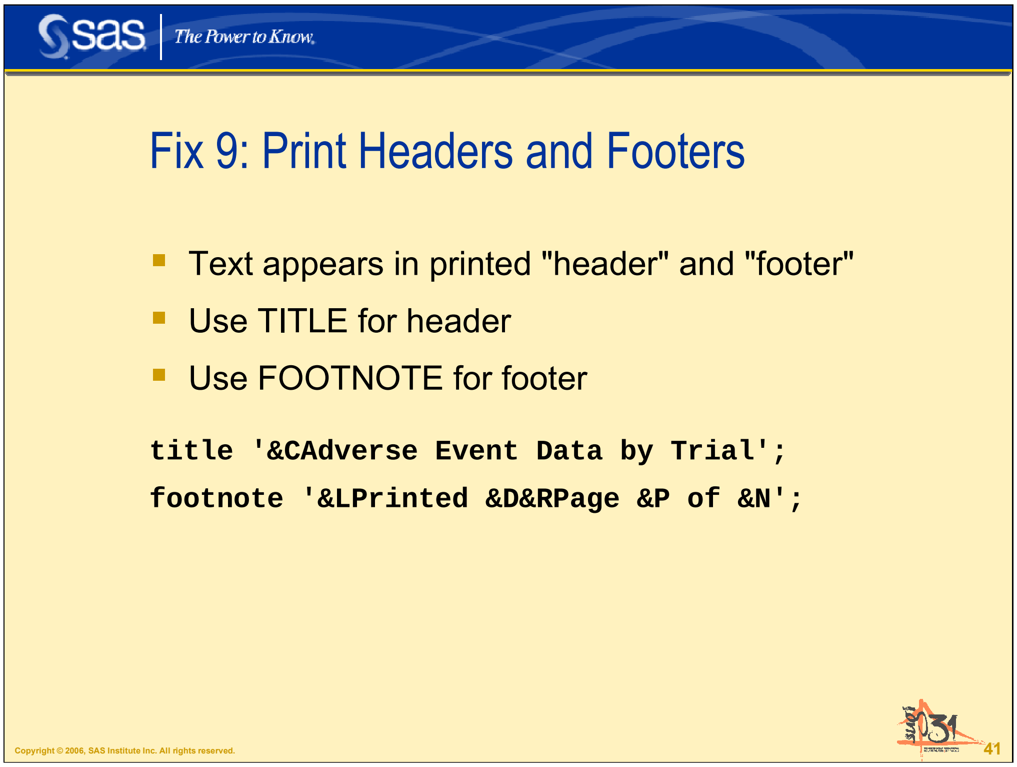
Page order

● Down, then over

○ Over, then down

If you click File > Page Setup from the Excel menu, you will be able to see that for the PRINT procedure output, the print orientation is set to "Landscape" and that rows 1 – 2 will repeat at top of each page.

If you follow the same steps for the TABULATE procedure output, you will find that the print orientation is set to "Portrait", and no rows will repeat at the top of each page.



The Power to Know.

Fix 9: Print Headers and Footers

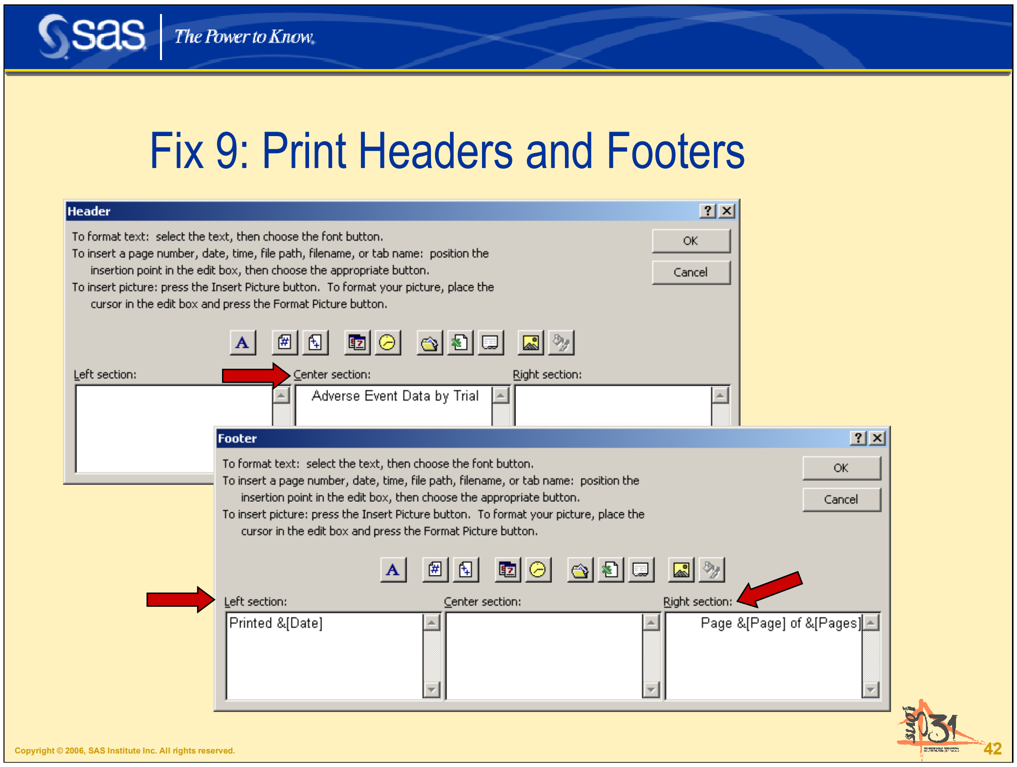
- Text appears in printed "header" and "footer"
- Use TITLE for header
- Use FOOTNOTE for footer

```
title '&CAdverse Event Data by Trial';
footnote '&LPrinted &D&RPage &P of &N';
```

Our final code change will to be add custom text that appears at the top and bottom of each printed page. By default, the ExcelXP tagset uses the text of TITLE statements as print headers, and the text of FOOTNOTE statements as print footers.

The TITLE statement shown above results in a print header with the text "Adverse Event Data by Trial" centered in the printed output. The special Excel control sequence "&C" causes all text following it to be centered.

The FOOTNOTE statement above is a bit more complicated. First, the word "Printed" will appear left-justified at the bottom of the printed output, followed by the current date. The "&D" control sequence causes the current date to be printed. Next, the text "Page *P* of *N*" will be right justified, where *P* is the current page number, and *N* is the page count.



Fix 9: Print Headers and Footers

To format text: select the text, then choose the font button.
To insert a page number, date, time, file path, filename, or tab name: position the insertion point in the edit box, then choose the appropriate button.
To insert picture: press the Insert Picture button. To format your picture, place the cursor in the edit box and press the Format Picture button.

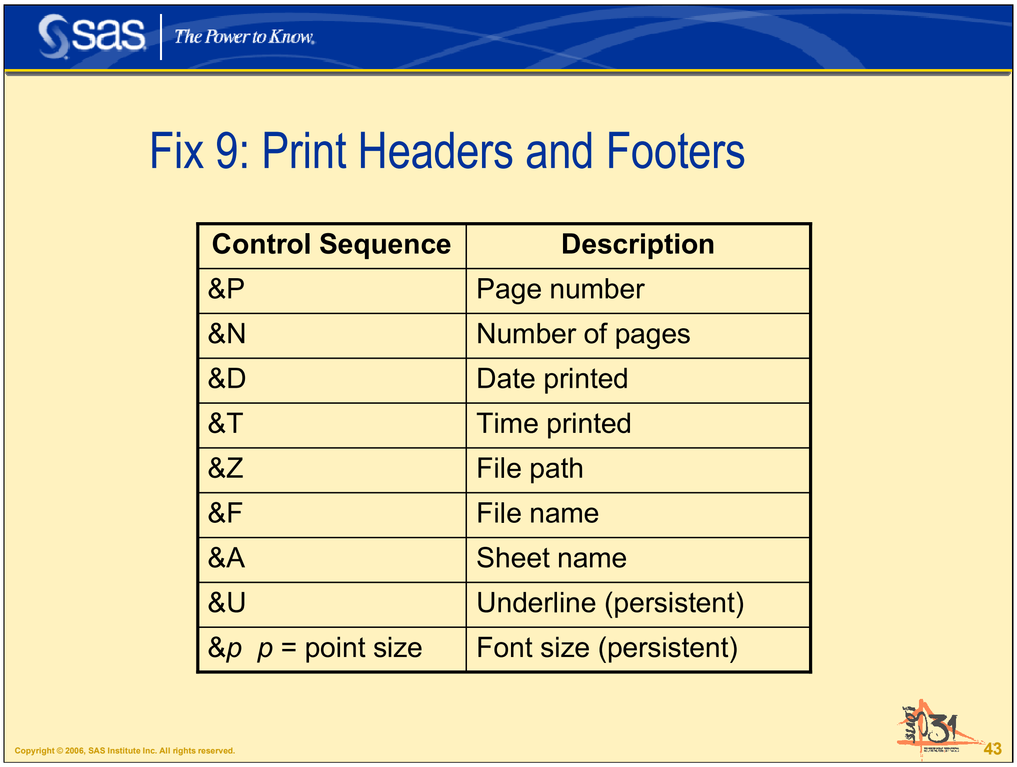
Left section:
Center section: Adverse Event Data by Trial
Right section:

To format text: select the text, then choose the font button.
To insert a page number, date, time, file path, filename, or tab name: position the insertion point in the edit box, then choose the appropriate button.
To insert picture: press the Insert Picture button. To format your picture, place the cursor in the edit box and press the Format Picture button.

Left section: Printed &[Date]
Center section:
Right section: Page &[Page] of &[Pages]

If you click File > Page Setup from the Excel menu, you can access dialogs that show the custom headers and footers for your workbook.

Note that we have centered text for the header. Also as expected, the date printed appears at the left of the footer and the page number and count appears at the right.



Fix 9: Print Headers and Footers

Control Sequence	Description
&P	Page number
&N	Number of pages
&D	Date printed
&T	Time printed
&Z	File path
&F	File name
&A	Sheet name
&U	Underline (persistent)
&*p* *p* = point size	Font size (persistent)

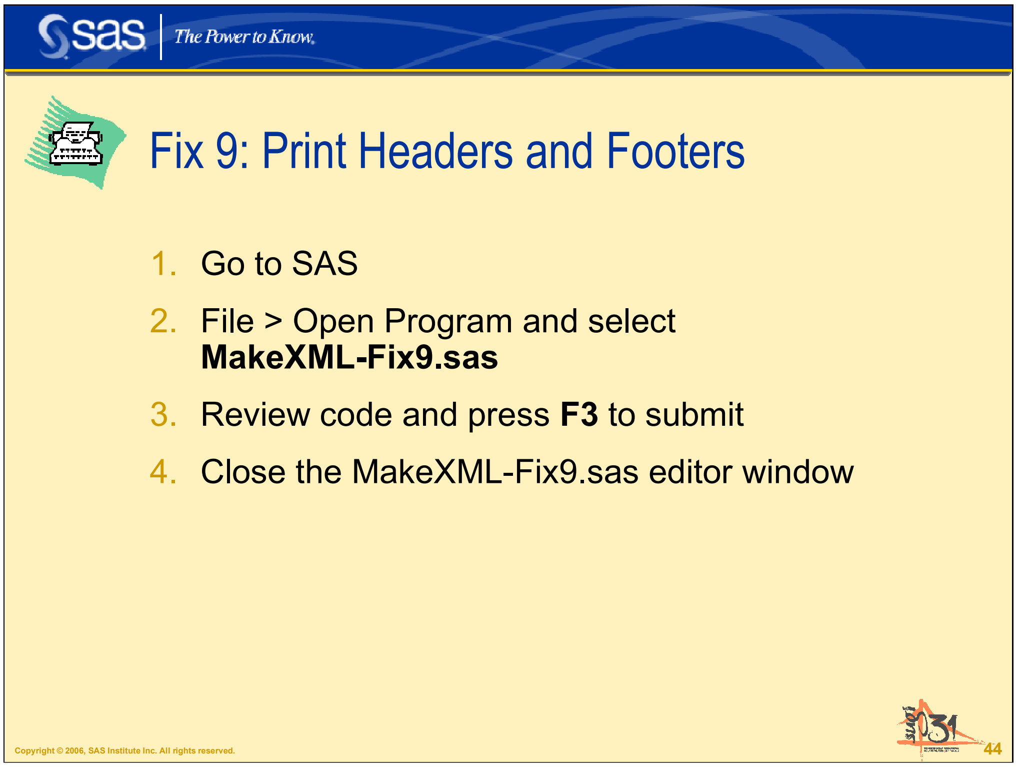
The table above shows some useful control sequences that you can use in print headers and footers. Note that the underline and font size sequences are persistent, just like Left, Center and Right justification. That is, once specified, they are in effect until changed.

For underlining just a portion of text, use:

```
This is the &Uunderlined&U text
```

To change the font size from 10 points to 18 points, then back to 10 points, use:

```
&10This is the &18BIG&10 text
```



Fix 9: Print Headers and Footers

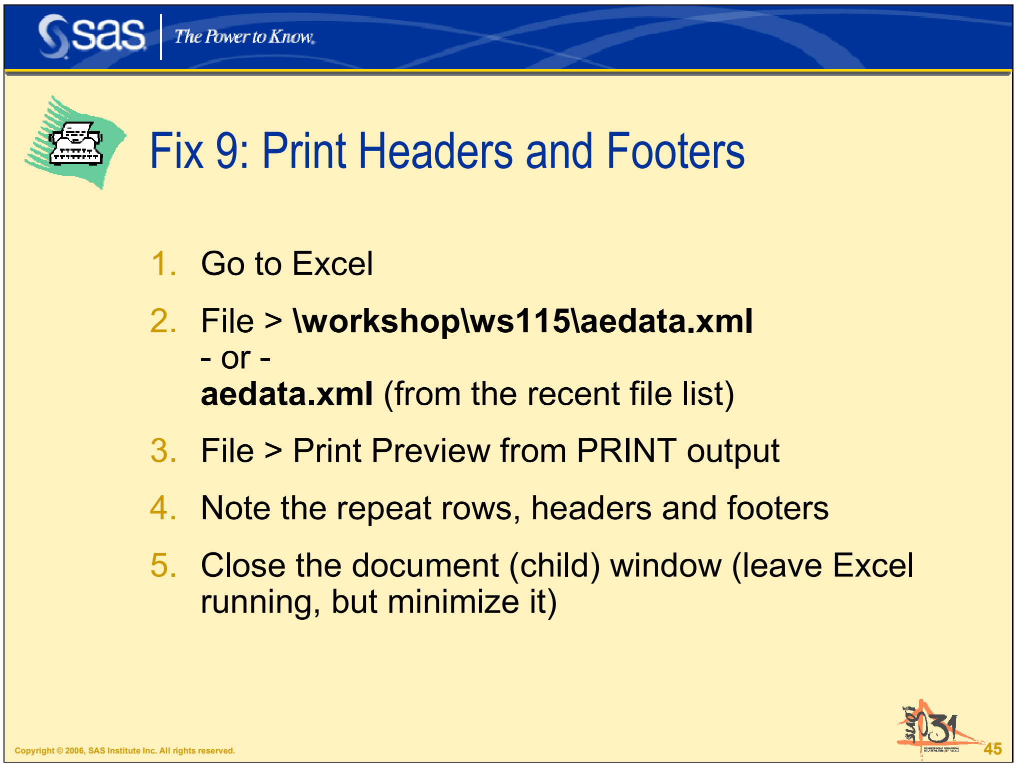
1. Go to SAS
2. File > Open Program and select **MakeXML-Fix9.sas**
3. Review code and press **F3** to submit
4. Close the MakeXML-Fix9.sas editor window

You do not need to make any changes to the code – they have been made for you.

In reviewing the code, you will notice that the following TITLE and FOOTNOTE statements were added:

```
title '&CAdverse Event Data by Trial';
footnote '&LPrinted &D&RPage &P of &N';
```

Finally, note that all the changes made up to this point have been included in this file: the correction for leading zeroes, custom worksheet names, corrected column widths for PRINT procedure output, corrected BY group labels, corrected column widths in the TABULATE procedure output, frozen column headings and AutoFilters for the PRINT procedure output, print orientation and repeating rows (PRINT procedure output only).



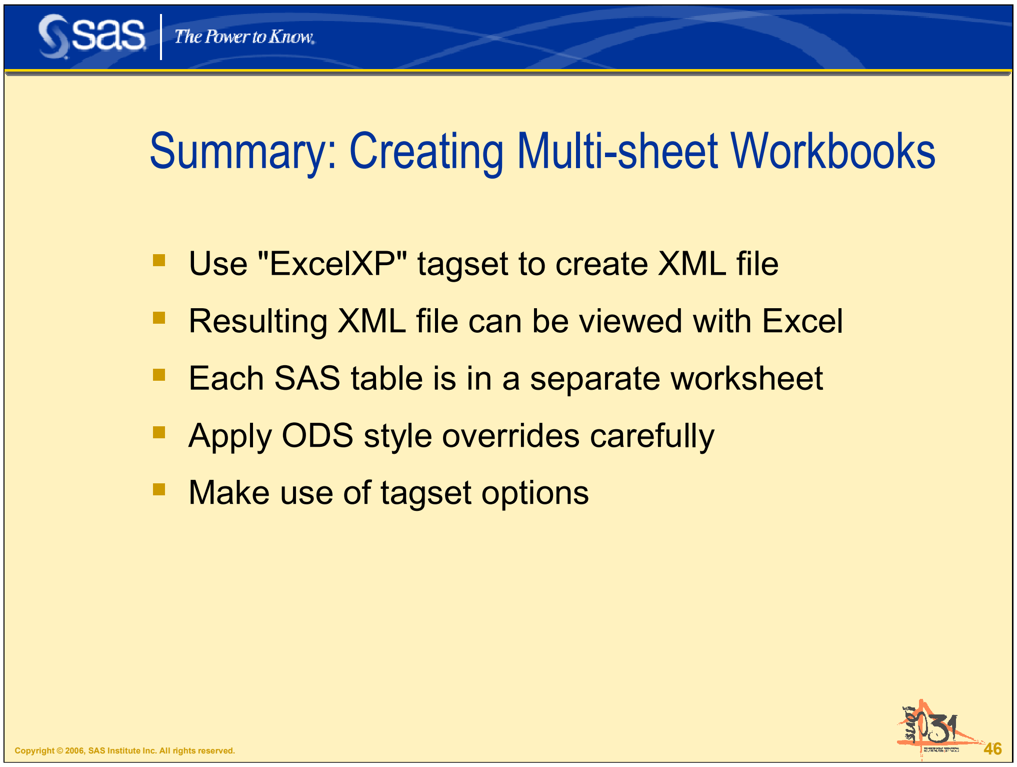
sas The Power to Know.

Fix 9: Print Headers and Footers

1. Go to Excel
2. File > **\workshop\ws115\aedata.xml**
   - or -
   aedata.xml (from the recent file list)
3. File > Print Preview from PRINT output
4. Note the repeat rows, headers and footers
5. Close the document (child) window (leave Excel running, but minimize it)

This is the final version of the Excel workbook.

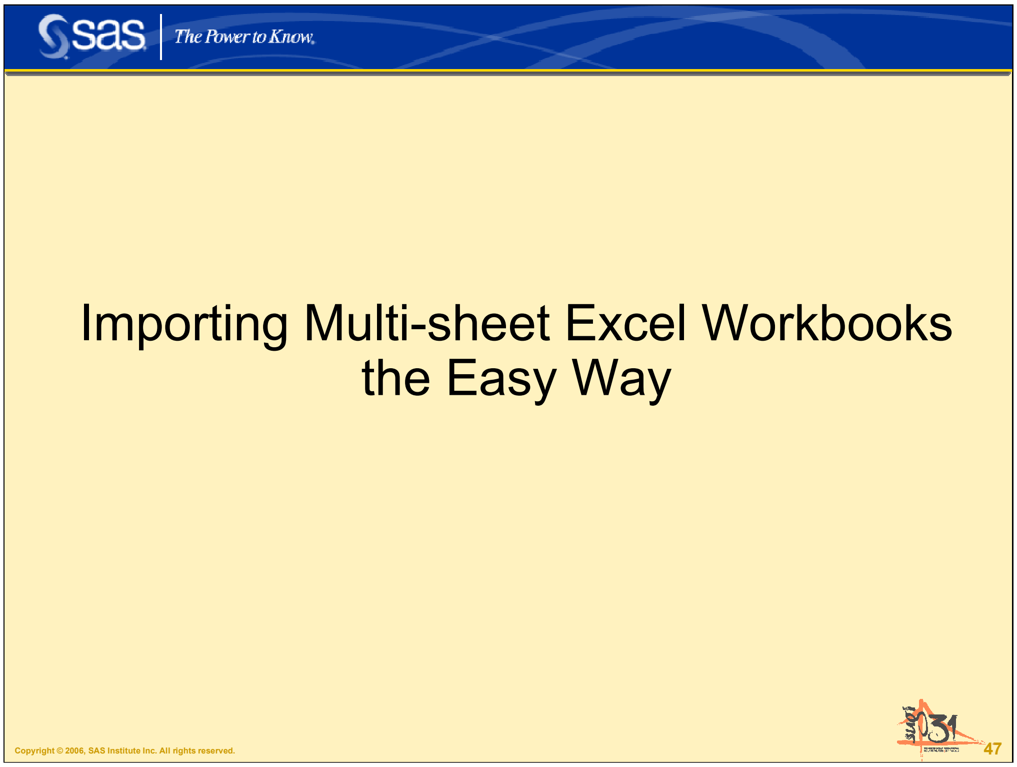
In Print Preview mode (File > Print Preview), you can see that the print headers and footers are in place, and also that rows 1-2 are repeated at the top of each page of the PROC PRINT output.



The Power to Know.

Summary: Creating Multi-sheet Workbooks

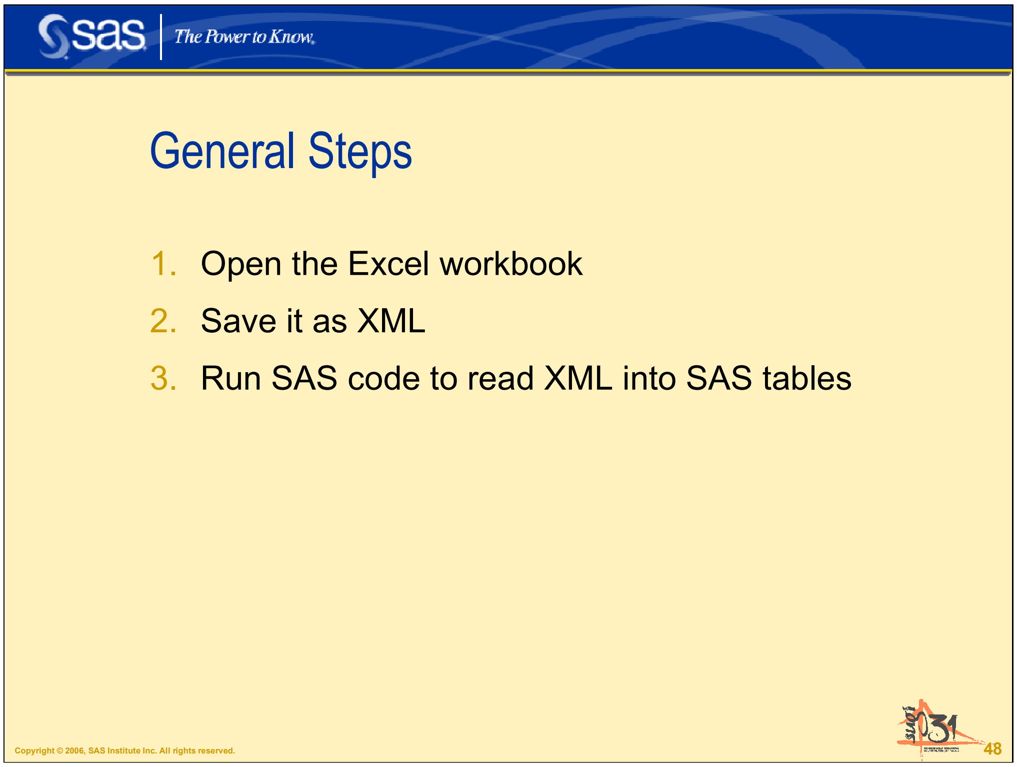
- Use "ExcelXP" tagset to create XML file
- Resulting XML file can be viewed with Excel
- Each SAS table is in a separate worksheet
- Apply ODS style overrides carefully
- Make use of tagset options



Importing Multi-sheet Excel Workbooks the Easy Way

This section focuses on importing Excel workbooks into SAS tables.

Excel XP (a.k.a. Excel 2002) or later is required for this operation because your Excel workbooks must first be saved as XML files. Only Excel XP and later support saving workbooks in XML format.

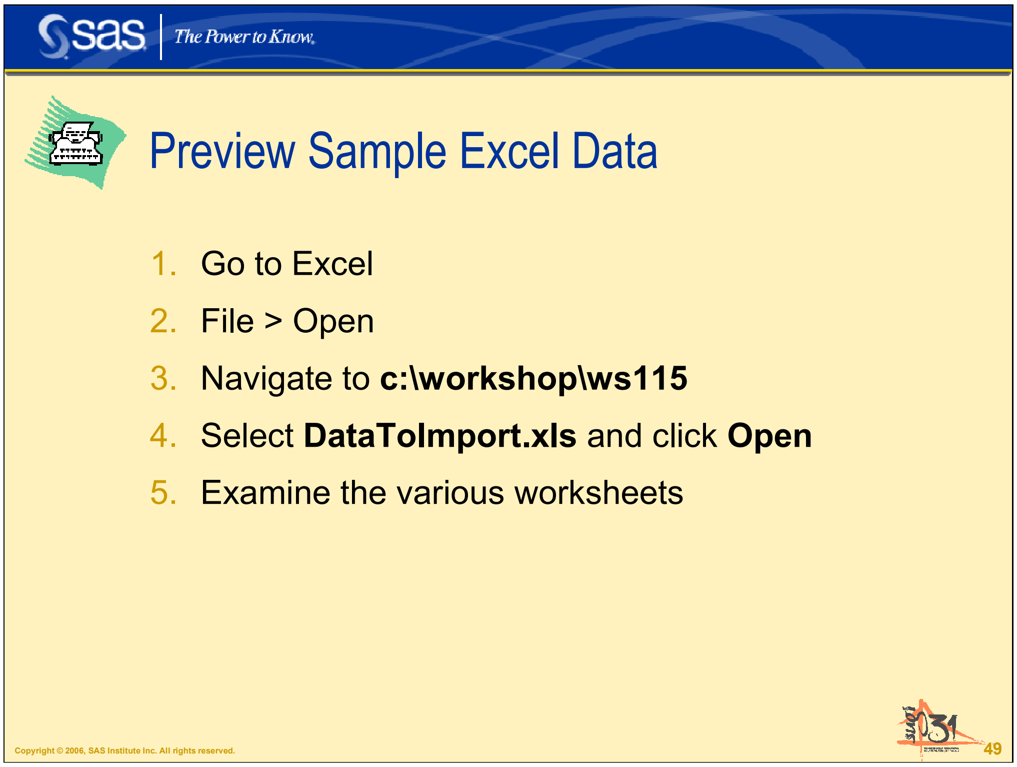


The Power to Know.

General Steps

1. Open the Excel workbook
2. Save it as XML
3. Run SAS code to read XML into SAS tables

We will use the new XML support features of SAS 9 to read in the Excel XML-format workbook into SAS tables. Specifically, we will use the SAS XML Libname Engine (SXLE) and a SAS XMLMap that was designed specifically for reading Excel XML data.

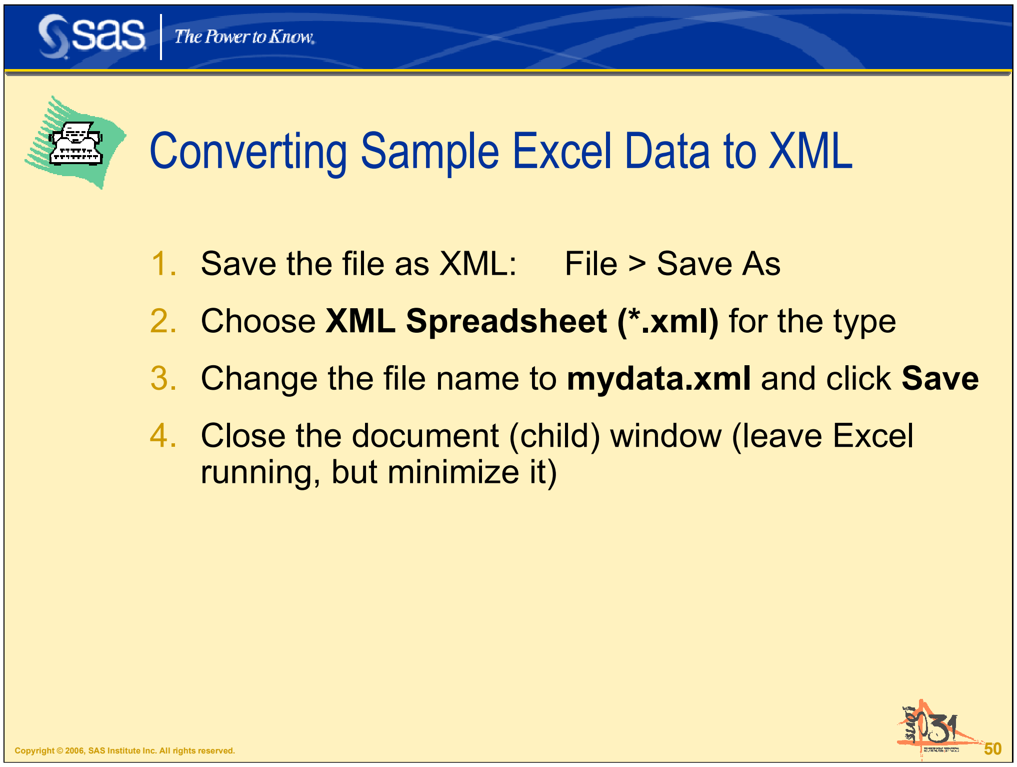


Preview Sample Excel Data

1. Go to Excel
2. File > Open
3. Navigate to **c:\workshop\ws115**
4. Select **DataToImport.xls** and click **Open**
5. Examine the various worksheets

This Excel workbook consists of 4 worksheets, each containing diverse data.

Each worksheet will be imported into a separate SAS table.

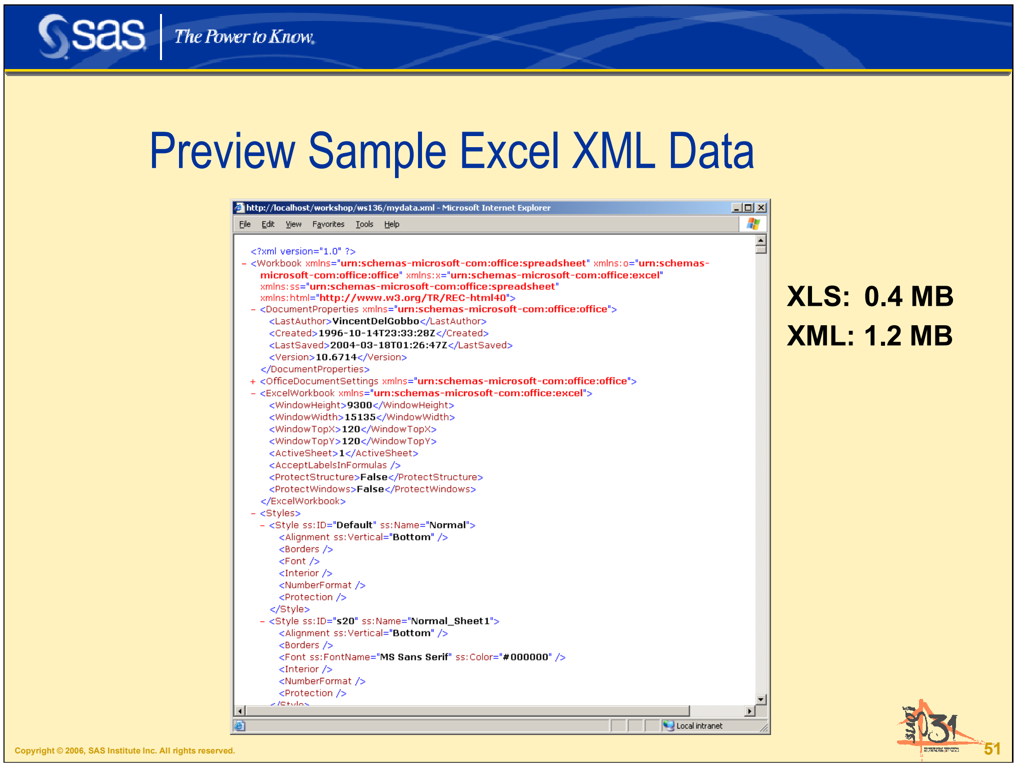


The Power to Know.

Converting Sample Excel Data to XML

1. Save the file as XML: File > Save As
2. Choose **XML Spreadsheet (*.xml)** for the type
3. Change the file name to **mydata.xml** and click **Save**
4. Close the document (child) window (leave Excel running, but minimize it)

The first step is to save the Excel Workbook as an XML file.



The Power to Know.

Preview Sample Excel XML Data

http://localhost/workshop/ws136/mydata.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help

```
  <?xml version="1.0" ?>
- <Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet" xmlns:o="urn:schemas-
    microsoft-com:office:office" xmlns:x="urn:schemas-microsoft-com:office:excel"
    xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
    xmlns:html="http://www.w3.org/TR/REC-html40">
  - <DocumentProperties xmlns="urn:schemas-microsoft-com:office:office">
      <LastAuthor>VincentDelGobbo</LastAuthor>
      <Created>1996-10-14T23:33:28Z</Created>
      <LastSaved>2004-03-18T01:26:47Z</LastSaved>
      <Version>10.6714</Version>
    </DocumentProperties>
  + <OfficeDocumentSettings xmlns="urn:schemas-microsoft-com:office:office">
  - <ExcelWorkbook xmlns="urn:schemas-microsoft-com:office:excel">
      <WindowHeight>9300</WindowHeight>
      <WindowWidth>15135</WindowWidth>
      <WindowTopX>120</WindowTopX>
      <WindowTopY>120</WindowTopY>
      <ActiveSheet>1</ActiveSheet>
      <AcceptLabelsInFormulas />
      <ProtectStructure>False</ProtectStructure>
      <ProtectWindows>False</ProtectWindows>
    </ExcelWorkbook>
  - <Styles>
    - <Style ss:ID="Default" ss:Name="Normal">
        <Alignment ss:Vertical="Bottom" />
        <Borders />
        <Font />
        <Interior />
        <NumberFormat />
        <Protection />
      </Style>
    - <Style ss:ID="s20" ss:Name="Normal_Sheet1">
        <Alignment ss:Vertical="Bottom" />
        <Borders />
        <Font ss:FontName="MS Sans Serif" ss:Color="#000000" />
        <Interior />
        <NumberFormat />
        <Protection />
```

XLS: 0.4 MB
XML: 1.2 MB

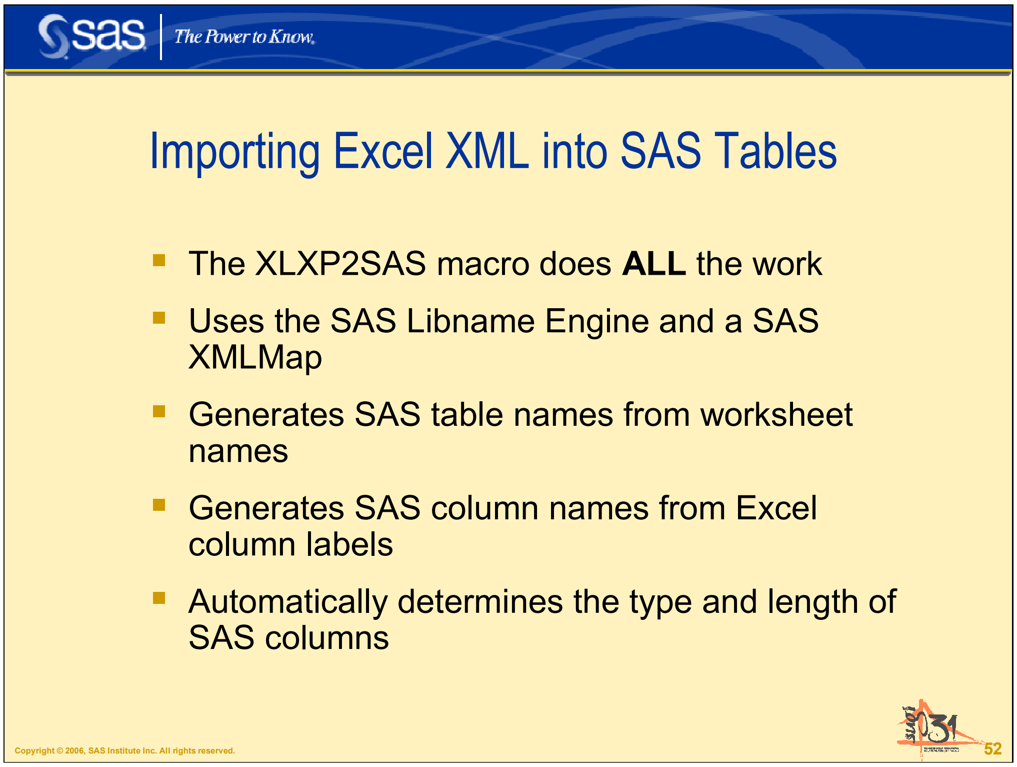
51

Next we will spend just a moment looking at the structure and size of the XML file that represents the Excel workbook.

If you wish, you can open Windows Explorer and navigate to the directory "c:\workshop\ws115" and double-click the file "mydata.xml". This will cause the file to open with Internet Explorer (by default) or whatever application you have defined to open XML files.

As you can see, this file is quite large, and writing DATA Step code to read it would be very cumbersome.

Fortunately, you are provided with a SAS XMLMap and macro (XLXP2SAS) that greatly simplifies the task of reading this XML data, *and any other Excel XML data*, into separate SAS tables.



Importing Excel XML into SAS Tables

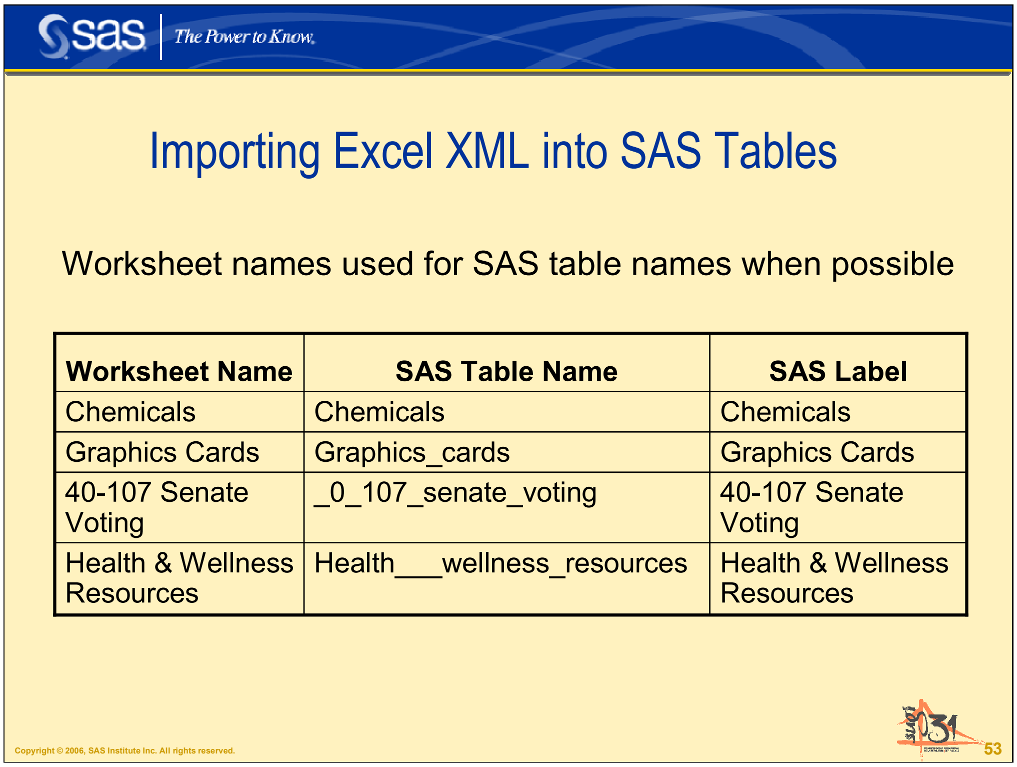
- The XLXP2SAS macro does **ALL** the work
- Uses the SAS Libname Engine and a SAS XMLMap
- Generates SAS table names from worksheet names
- Generates SAS column names from Excel column labels
- Automatically determines the type and length of SAS columns

The XLXP2SAS macro is provided for you, and does all the work of importing the Excel XML file into a SAS table. The macro uses the SAS Libname Engine and a SAS XMLMap, both of which are part of Base SAS.

The purpose of a SAS XMLMap is to map generic XML elements to SAS columns. It is used in conjunction with the SAS Libname Engine to import an XML files into a SAS table. An Excel-specific XMLMap has been developed and provided for you to use with the XLXP2SAS macro – you do not have to write any code.

The worksheet names will be used for the SAS table names and similarly, SAS column names will be generated from the Excel column labels. If no Excel column labels are available, then the SAS column names will be "COLUMN1", "COLUMN2", etc.

Only Base SAS is required to take advantage of this code, and it works for any XML file generated by Excel.



The Power to Know.

Importing Excel XML into SAS Tables

Worksheet names used for SAS table names when possible

Worksheet Name	SAS Table Name	SAS Label
Chemicals	Chemicals	Chemicals
Graphics Cards	Graphics_cards	Graphics Cards
40-107 Senate Voting	_0_107_senate_voting	40-107 Senate Voting
Health & Wellness Resources	Health___wellness_resources	Health & Wellness Resources

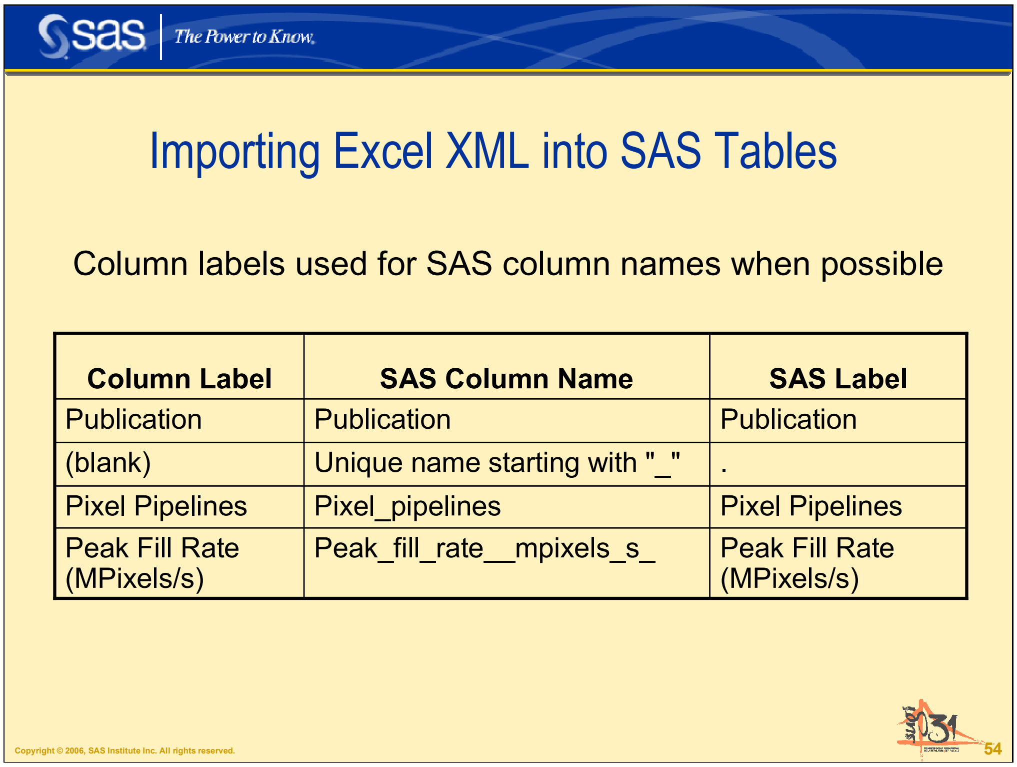
The table above shows how table names and labels will be derived from our sample XML file.

When the worksheet name does not contain any spaces or invalid characters, as is the case with "Chemicals", that name will be used as the SAS table name.

Otherwise, all invalid characters will be converted to underscores, and the resulting value will be used for the SAS table name.

Finally, the original worksheet name will be used as the label for the SAS table.

In all cases, the SAS table name will be truncated to 32 characters, and the SAS label will be truncated to 256 characters.



Importing Excel XML into SAS Tables

Column labels used for SAS column names when possible

Column Label	SAS Column Name	SAS Label
Publication	Publication	Publication
(blank)	Unique name starting with "_"	.
Pixel Pipelines	Pixel_pipelines	Pixel Pipelines
Peak Fill Rate (MPixels/s)	Peak_fill_rate__mpixels_s_	Peak Fill Rate (MPixels/s)

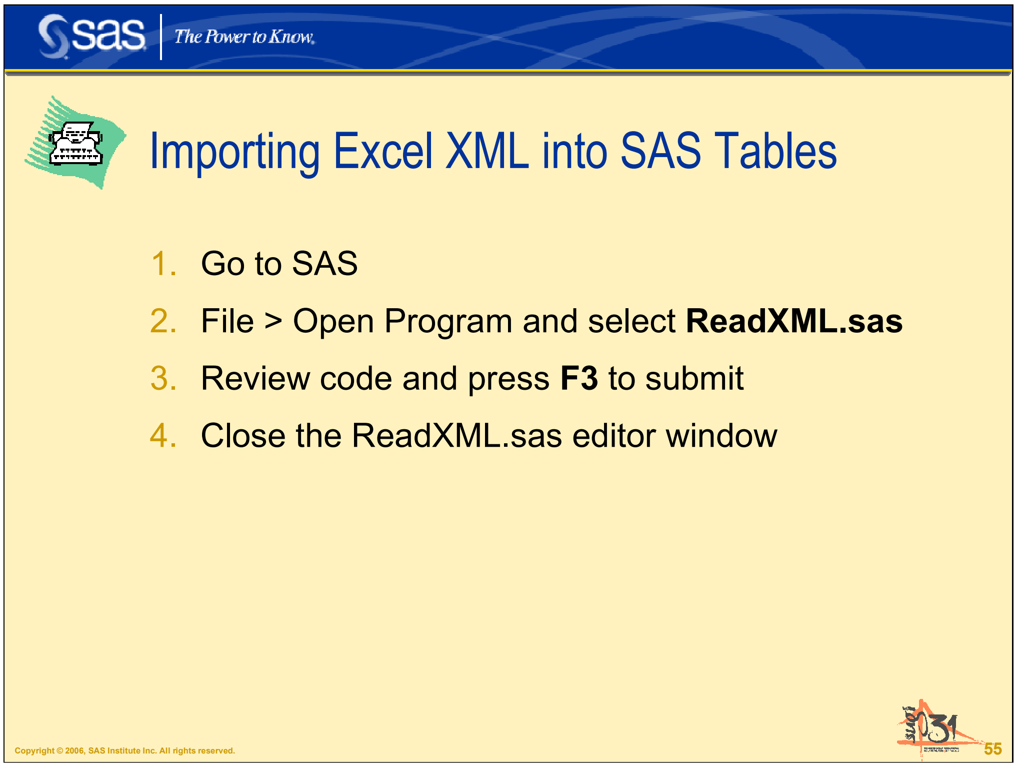
The table above shows an example of how SAS column names and labels will be derived from our sample XML file.

When the worksheet column label does not contain any spaces or invalid characters, as is the case with "Publication", that label will be used as the SAS column name.

Otherwise, all invalid characters will be converted to underscores, and the resulting value will be used for the SAS column name.

Finally, the original worksheet column label will be used as the label for the SAS table column.

In all cases, the SAS column name will be truncated to 32 characters, and the column label will be truncated to 256 characters.



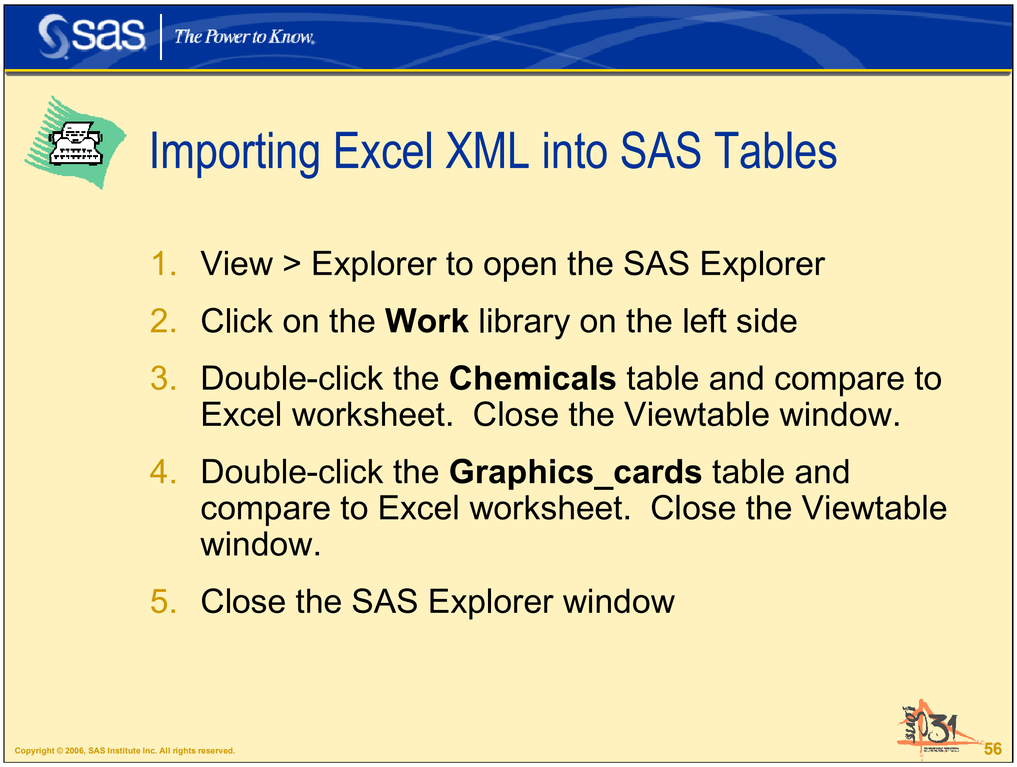
Importing Excel XML into SAS Tables

1. Go to SAS
2. File > Open Program and select **ReadXML.sas**
3. Review code and press **F3** to submit
4. Close the ReadXML.sas editor window

The XLXP2SAS macro was made available to your SAS session using a %INCLUDE statement in the file "Setup.sas". In a production environment, you may wish to add the XLXP2SAS macro to your autocall macro library.

Run the XLXP2SAS macro via the program "ReadXML.sas". You only need to specify two arguments: the location of the XML file you wish to import, and the location of the SAS XMLMap (which is provided for you).

The macro reads the huge XML file with the help of the SAS Libname Engine and the SAS XMLMap. In our case, the result is 4 tables stored in the WORK library.



sas The Power to Know.

Importing Excel XML into SAS Tables

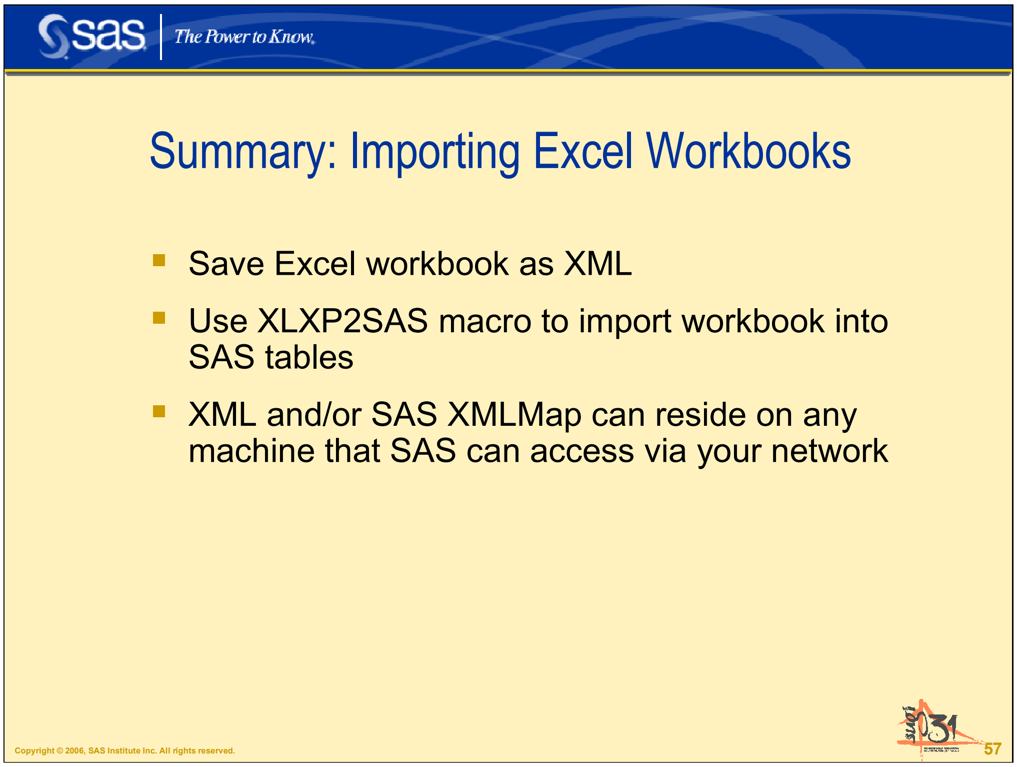
1. View > Explorer to open the SAS Explorer
2. Click on the **Work** library on the left side
3. Double-click the **Chemicals** table and compare to Excel worksheet. Close the Viewtable window.
4. Double-click the **Graphics_cards** table and compare to Excel worksheet. Close the Viewtable window.
5. Close the SAS Explorer window

Chemicals table

- SAS column labels match the Excel worksheet column labels
- The column "CAA 112(r) TQ" is correctly typed as numeric
- All other columns correctly typed as character

Graphics_cards table

- SAS column labels match the Excel worksheet column labels
- The SAS columns "Pixel Pipelines" and "Memory bus width (bits)" are correctly typed as character.
- All other columns correctly typed as numeric



Summary: Importing Excel Workbooks

- Save Excel workbook as XML
- Use XLXP2SAS macro to import workbook into SAS tables
- XML and/or SAS XMLMap can reside on any machine that SAS can access via your network

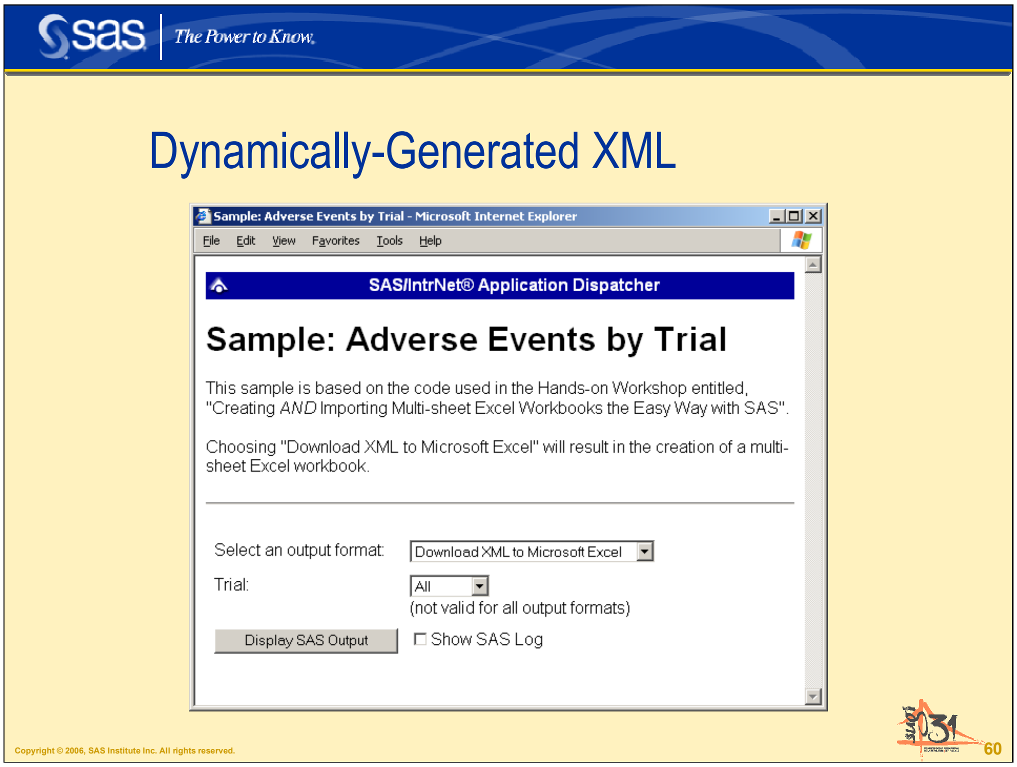
Using SAS/IntrNet

and

Stored Processes

SAS/IntrNet and Stored Processes

- SAS code is run from non-SAS client
- SAS is on any platform
- Client needs only a Web browser
- SAS output is delivered in "real time"
- Web-enable the code we've been using



The Power to Know.
Sample: Adverse Events by Trial - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help

Dynamically-Generated XML

SAS/IntrNet® Application Dispatcher

Sample: Adverse Events by Trial

This sample is based on the code used in the Hands-on Workshop entitled, "Creating *AND* Importing Multi-sheet Excel Workbooks the Easy Way with SAS".

Choosing "Download XML to Microsoft Excel" will result in the creation of a multi-sheet Excel workbook.

Select an output format: Download XML to Microsoft Excel

Trial: All
(not valid for all output formats)

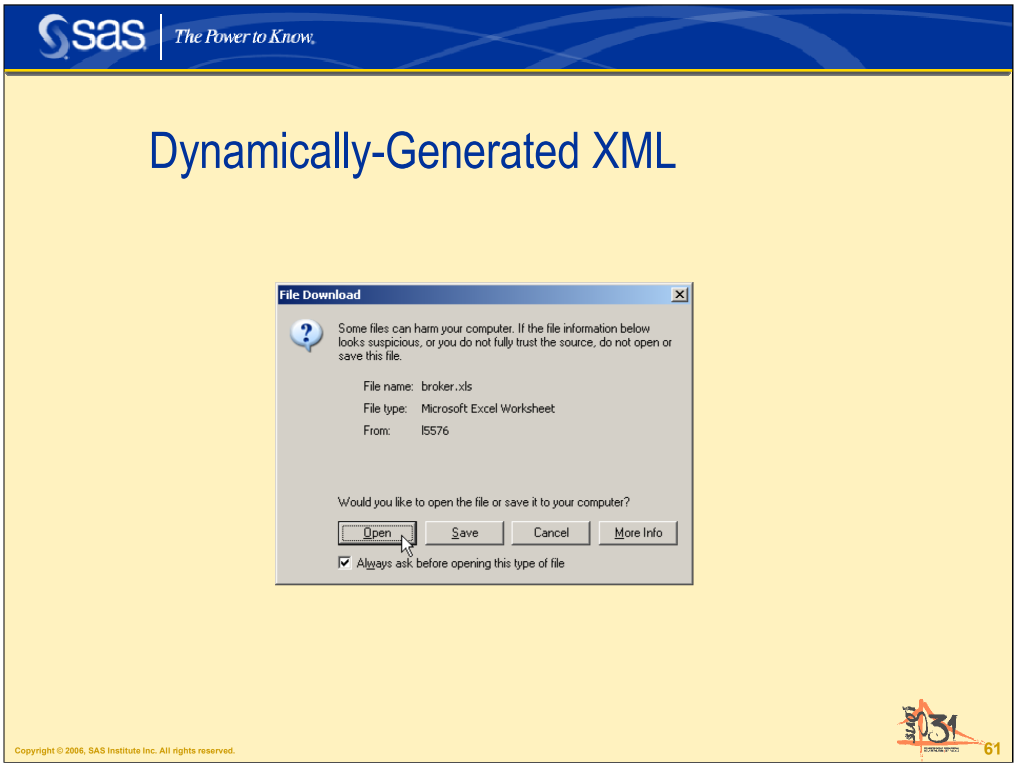
Display SAS Output ☐ Show SAS Log

Here is a simple Web page that can be used to execute SAS code stored on a server using the SAS/IntrNet product.

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

The image above indicates that we will be generating XML for use with Microsoft Excel.

Clicking "Display SAS Output" sends the choices made in the Web page to the SAS server as global macro variables. Those macro variables are used by the SAS program to control the type of output generated.



Dynamically-Generated XML

File Download

Some files can harm your computer. If the file information below looks suspicious, or you do not fully trust the source, do not open or save this file.

File name: broker.xls
File type: Microsoft Excel Worksheet
From: l5576

Would you like to open the file or save it to your computer?

☑ Always ask before opening this type of file

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

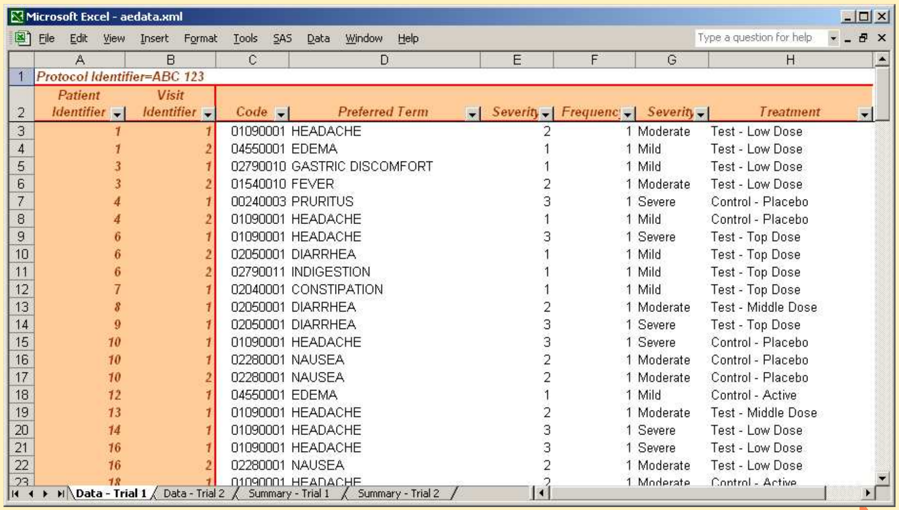
Note that you are presented with an Open/Save dialog, instead of the results being displayed in the Web browser. Clicking Open will cause the SAS output to be displayed in Excel, provided that Excel is installed on the machine.

This SAS/IntrNet-specific code, which must be specified before any ODS statements, was used to cause the SAS output to be displayed in Excel:

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

This code will also work with SAS Stored Processes.

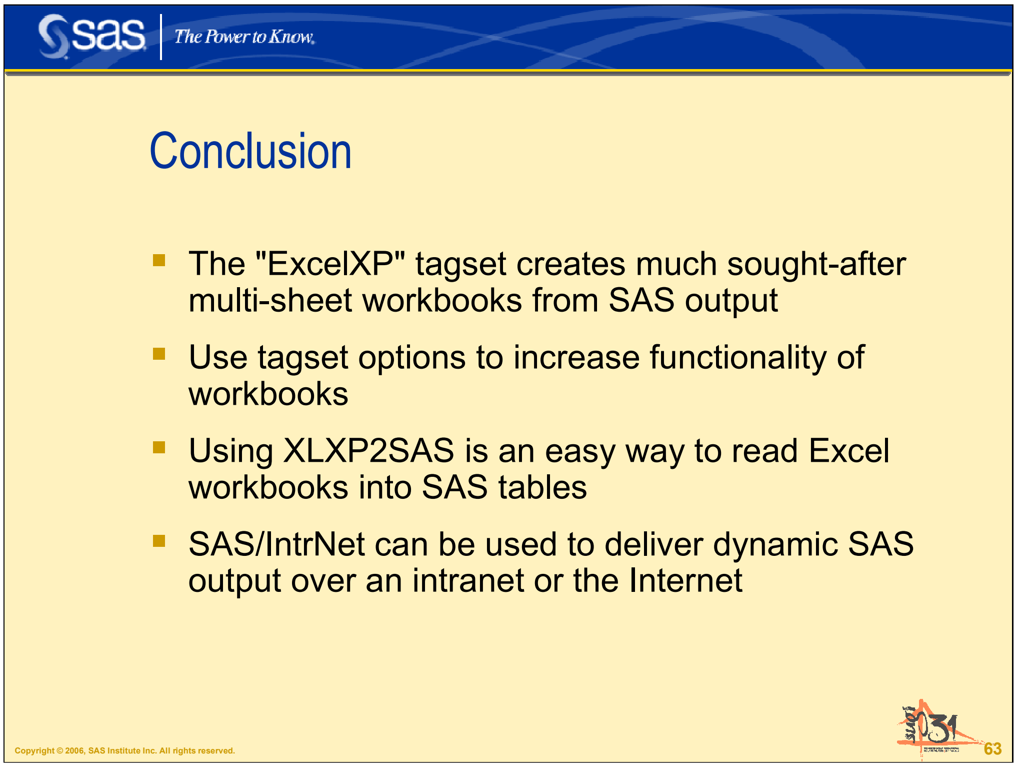
Dynamically-Generated XML



Protocol Identifier=ABC 123

Patient Identifier	Visit Identifier	Code	Preferred Term	Severity	Frequenc	Severity	Treatment
1	1	01090001	HEADACHE	2	1	Moderate	Test - Low Dose
1	2	04550001	EDEMA	1	1	Mild	Test - Low Dose
3	1	02790010	GASTRIC DISCOMFORT	1	1	Mild	Test - Low Dose
3	2	01540010	FEVER	2	1	Moderate	Test - Low Dose
4	1	00240003	PRURITUS	3	1	Severe	Control - Placebo
4	2	01090001	HEADACHE	1	1	Mild	Control - Placebo
6	1	01090001	HEADACHE	3	1	Severe	Test - Top Dose
6	2	02050001	DIARRHEA	1	1	Mild	Test - Top Dose
6	2	02790011	INDIGESTION	1	1	Mild	Test - Top Dose
7	1	02040001	CONSTIPATION	1	1	Mild	Test - Top Dose
8	1	02050001	DIARRHEA	2	1	Moderate	Test - Middle Dose
9	1	02050001	DIARRHEA	3	1	Severe	Test - Top Dose
10	1	01090001	HEADACHE	3	1	Severe	Control - Placebo
10	1	02280001	NAUSEA	2	1	Moderate	Control - Placebo
10	2	02280001	NAUSEA	2	1	Moderate	Control - Placebo
12	1	04550001	EDEMA	1	1	Mild	Control - Active
13	1	01090001	HEADACHE	2	1	Moderate	Test - Middle Dose
14	1	01090001	HEADACHE	3	1	Severe	Test - Low Dose
16	1	01090001	HEADACHE	3	1	Severe	Test - Low Dose
16	2	02280001	NAUSEA	2	1	Moderate	Test - Low Dose
18	1	01090001	HEADACHE	2	1	Moderate	Control - Active

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.



The Power to Know.

Conclusion

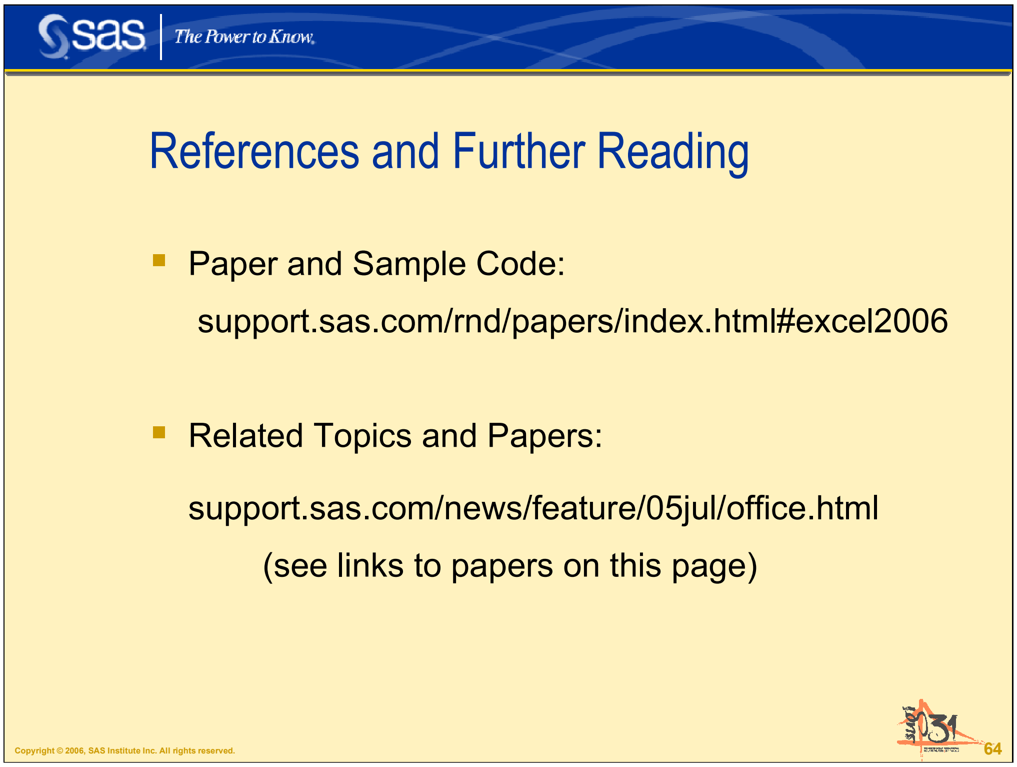
- The "ExcelXP" tagset creates much sought-after multi-sheet workbooks from SAS output
- Use tagset options to increase functionality of workbooks
- Using XLXP2SAS is an easy way to read Excel workbooks into SAS tables
- SAS/IntrNet can be used to deliver dynamic SAS output over an intranet or the Internet

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

The SAS 9.1 "ExcelXP" tagset complies with the Microsoft XML Spreadsheet Specification and provides an easy way to export your data to Excel workbooks that contain multiple worksheets. The tagset supports a great number of options that can be used provide many Excel functions.

While it may be a bit cumbersome to use the SXLE and XMLMaps to import Excel data to SAS, the use of the XLXP2SAS macro greatly simplifies the task.

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 Office applications.



References and Further Reading

- Paper and Sample Code:

  support.sas.com/rnd/papers/index.html#excel2006

- Related Topics and Papers:

  support.sas.com/news/feature/05jul/office.html

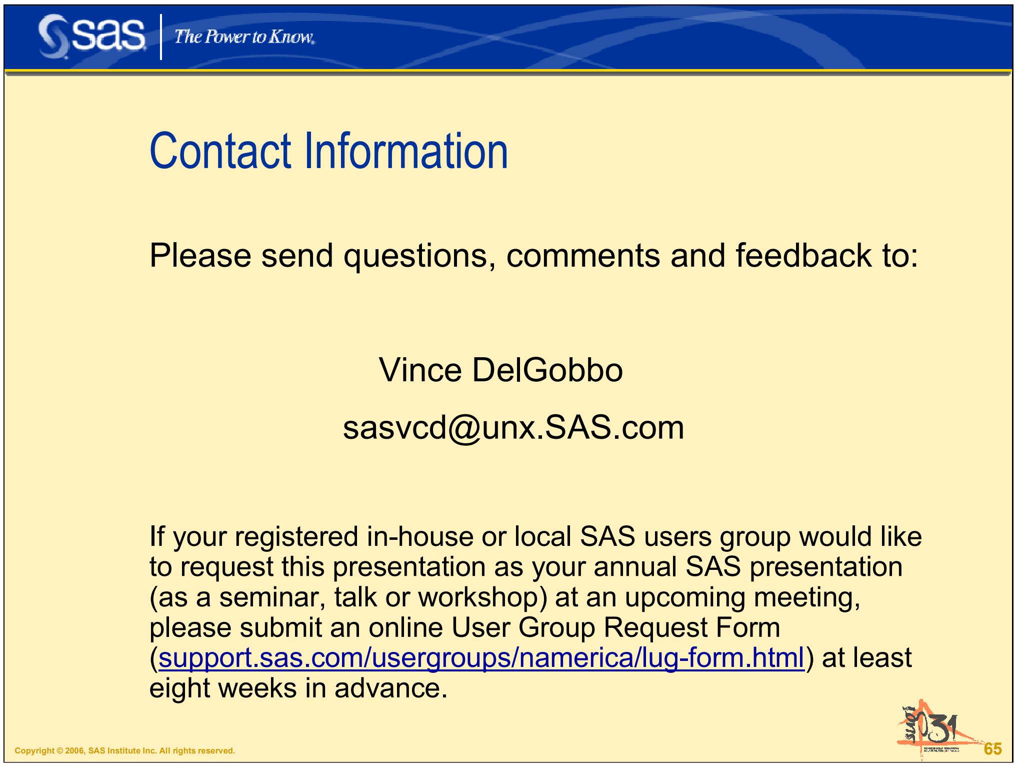
  (see links to papers on this page)

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#excel2006 and find the entry "Creating *AND* Importing Multi-Sheet Excel Workbooks the Easy Way with SAS".

IMPORTANT NOTE: Review the "Installation" and "Usage" sections of the file named "ReadMe.txt" for important information on using the sample files.

Other related information can be found at:
http://support.sas.com/news/feature/05jul/office.html

Refer to the ODS documentation for your release of SAS to find out more about styles, style attributes and the TEMPLATE procedure.



Contact Information

Please send questions, comments and feedback to:

Vince DelGobbo

sasvcd@unx.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.

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.