# An Optimal Way to Import Excel® Worksheets into PC SAS®

Nathaniel Derby, Statis Pro Data Analytics, Seattle, WA

## ABSTRACT

There are many ways to import data from Microsoft® Excel® into PC SAS®, and the "optimal" way often depends upon the project and the programmer's preference. However, by adhering to clear definitions of three generally desirable properties (precision, flexibility and automation), we can agree upon a definition of "optimal". We then propose a solution that fulfills these three properties. This is most appropriate for situations where data from a large set of Excel worksheets is periodically loaded into SAS, where each Excel worksheet has the same structure from load to load (although the worksheets can have different structures from each other). This proposed solution involves an easily used macro which works on all versions of PC SAS, Windows® and Excel, and which is freely available for downloading.

Keywords: SAS, DDE, Excel, Export, Import, X4ML.

This paper is an excerpt[1] from Derby (2008b), which may be updated and can be downloaded for more information. The macro described here and all examples in this paper can also be downloaded from the project website listed at the end of this paper.

## INTRODUCTION

### IMPORTING EXCEL DATA INTO SAS

While SAS can accept input from a variety of formats (see, e.g., Delwiche and Slaughter (2003, ch.2)), Excel files (XLS) are among the most common. Currently there are many ways to import Excel data into SAS: The Import Wizard, `PROC IMPORT`, using a `LIBNAME` statement, Dynamic Data Exchange, or by saving the Excel worksheets as CSV or TXT files that can be read by SAS. Which is the "best" way?

For much of the time, the input process is dictated by the project, the preferences/abilities of the programmer, or inheritance ("the way it's always been done"). These are certainly common and understandable situations. However, for the programmer who is either starting from scratch or who has a chance to change an existing project for the better, it may be worth investgating how best to go about this problem. Indeed, when small projects become larger ones, small inefficiencies can become large ones: Using the import wizard once per month is quite different from using it 20 times daily!

There are three criteria we can use for choosing the "best" method;

- *Precision:* Are we sure to read the input data correctly? Since Excel does not require its columns to have a consistent format, we can have character data (like "NA" or "-") in a numeric column, thus causing automated SAS procedures like `PROC IMPORT` to either missclassify the entire column as character data, or to missclassify a value like "-" (which often designates zero) as a missing value – in both cases without a warning or error message to alert the user about a possible data problem. Often the programmer knows the data informats better than SAS's algorithms for determining it. Therefore, this should be hard-coded in the input program.
- *Flexibility:* Some methods, such as the macros of Sun and Wong (2005) or Rashleigh-Berry (2008), have limitations which can impede their usefulness;[2] E.g., Sun and Wong's macro does not allow for many special characters to be in any worksheet name, and Rashleigh-Berry's does not allow for spaces in either the path or name of the Excel file. Furthermore, both of them export data into SAS only as character variables, thus requiring another SAS program to convert the data into the proper formats. For maximal usefulness, we should use a flexible method.
- *Automation:* Here we simply mean that everything is done within the SAS program, so that the programmer need only run the code. If the programmer first needs to save each Excel spreadsheet as a CSV or TXT file, or run an import wizard, this is not automated.

With these criteria, we *almost* have a solution: To read a CSV or TXT file extracted from the Excel worksheet using a `DATA` step, as shown in Delwiche and Slaughter (2003, pp. 58-59):

```
DATA reading;
  INFILE 'c:\MyRawData\Books.txt' DLM = '09'x;
  INPUT Name $ Week1 Week2 Week3 Week4 Week5;
```

[1] Reprinted with permission of the author.

[2] Qi (2004) describes another macro for exporting data from Excel into SAS - and that macro's major limitation is that it is not available to the public!

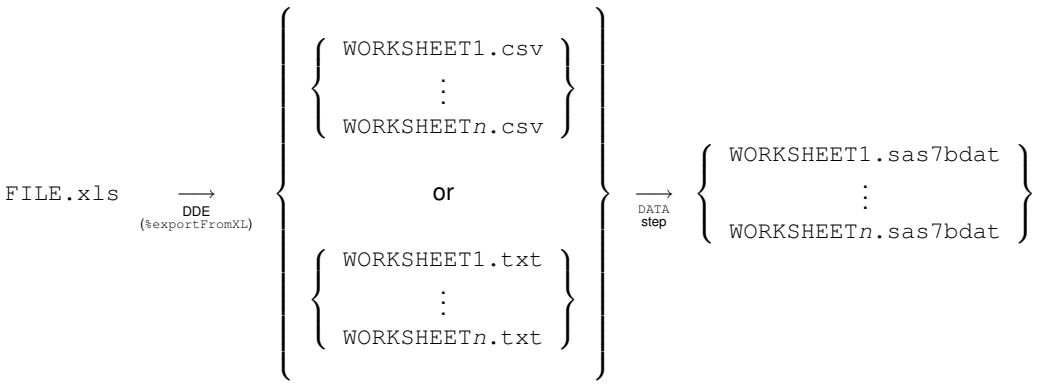



Figure 1: Flow chart for data from an Excel file into SAS. Naturally, the names of the SAS data sets (`*.sas7bdat`) can be given other names.

```
DATA music;
  INFILE 'c:\MyRawData\Bands.csv' DLM = ',' DSD MISSOVER;
  INPUT BandName :$30. GigDate :MMDDYY10. EightPM NinePM TenPM ElevenPM;
RUN;
```

This approach fulfills the first two criteria:

- *Precision:* Since we are explicitly giving the variable types and/or formats (presumeably based on the programmer's knowledge of the data), we are regaining control from SAS' algorithms over the precision of our variables.
- *Flexibility:* Since we are writing the SAS code ourselves, we can clearly modify it to fit whatever structure the data has. Furthermore, for reading from files with slight variations in the data structure, using the macro language can help - see Example 4 or Derby (2008b) for details.

At this point, there is absolutely nothing new in the method proposed in this paper. Indeed, these are major reasons why it is so common to read from CSV or TXT files into SAS. However, we do fall short in our last criterion:

- *Automation:* This approach is not automated in that we must first save our input Excel file as a CSV or TXT file. While this may seem trivial, it becomes critical in the case of many (e.g., over 50) spreadsheets of several files, especially when there is also time pressure.

The premise of this paper is to use a macro (`%exportFromXL`) which uses Dynamic Data Exchange to save each worksheet in a given Excel file as either a CSV or a TXT file – after which the data can be read by a simple SAS `DATA` step. This data flow is illustrated in Figure 1. The details are given below.

## DYNAMIC DATA EXCHANGE

Dynamic Data Exchange (DDE), as explained in Vyverman (2001), Watts (2005) and Derby (2008a), is a practical, effective and reliable solution to the automation part of this problem. The defining characteristic of DDE is that SAS operates the interface of a PC application directly, using commands written in the application native language within a `DATA _NULL_` step. While DDE can be used between SAS and any Windows program (e.g., Word® (Vyverman, 2003) and PowerPoint® (Vyverman, 2005)), most of its use is with Excel, with commands written in X4ML (the predecessor to VBA).

Unlike in situations where DDE is used to export data *from* SAS *into* Excel (Vyverman, 2000, 2001; Watts, 2004, 2005; Derby, 2008a), where DDE does all the work, here DDE represents just a portion of the solution. Precisely, DDE is used to export the Excel data into CSV or TXT files, after which regular SAS code can be used to finish importing the data into SAS. While it may seem strange to use DDE for just part of the solution, this is quite common in practice. Indeed, Watts (2005) details how DDE can effectively be used to automate small tasks, and Van Campen (2007) uses DDE after creating XML files to convert them into native Excel (XLS) files – an approach that can also be used with the ExcelXP tagset. DDE use for small tasks isn't just relegated to SAS projects; WinEdt™ (Simonic, 2008), the software used to create this document, uses DDE at the very end of its text processing to display the updated output within Adobe Reader®.

While DDE is used in these applications to perform small tasks, it should be emphasized that these are also *important* tasks, which would be onerous to do manually – especially if done repeatedly. Likewise, in this situation, we are using DDE to automate the small (but important) task of exporting Excel data into CSV or TXT files.

## USAGE

### INSTALLATION

The macro introduced here, %exportFromXL, is actually a group of smaller component macros, as explained in Derby (2008b). To use it within a SAS program, put these macros (all found in the *exportFromXL* directory of the unzipped file from the project download[3]) into a central directory – e.g., c:\SAS\exportFromXL. If the Excel application is in a language other than English, open the file exportFromXL.sas and change the language by changing the default parameter value lang=en to lang=xx, where xx is the appropriate language code.[4] Then add the following[5] to the SAS program in question (usually at the heading) before calling %exportFromXL:

```
%LET froot = c:\SAS\exportFromXL;

OPTIONS SASAUTOS=( "&froot" ) MAUTOSOURCE MCOMPILENOTE=all NOTES SOURCE SOURCE2;
```

This tells SAS to look at the contents of the &imroot directory to find new macro definitions (in particular, %exportFromXL and its component macros). We could avoid the %let statement altogether and write sasautos=( 'c:\SAS\exportFromXL' ) in the options statement instead, but the above solution is preferable for program portability. For each example in this paper, we define an input root (inroot) for the imported Excel files and an output root (outroot) for our outputted CSV or TXT files:

```
%LET inroot = c:\SAS\Example xx;
%LET outroot = c:\SAS\Example xx;
```

For simplicity, we have made these the same in the examples (downloadable from the project website).

### PARAMETERS

%exportFromXL is based on the %exportToXL macro by Derby (2007) and has a similar structure. Here we explain its usage and provide examples – a discussion of the code itself can be found in Derby (2008b). Some of the parameters might best be understood via the examples that follow.

```
%exportFromXL( inpath = ,        inname = ,           outpath = ,   filetype = csv,   delimiter = comma,
               sheets = [ALL],   skipsheets = [NONE], lang = en,    endclose = yes );
```

- inpath/inname: The full directory path (inpath) and name (inname) of the Excel file whose worksheets will be saved as CSV or TXT files.
- outpath: The full directory path of the CSV or TXT files that will be made from the Excel worksheets. The names will be taken from the names of the worksheets.
- filetype: The type of file that will be the output - csv or txt.
- delimiter: comma or tab. (Not all TXT files are tab-delimited)
- sheets: The names of the worksheets of the Excel input which will be outputted into CSV or TXT, separated by colons (:). This is done so that we can accommodate worksheet names that include commas or spaces. The default ([ALL]) means that all included worksheets will be exported.[6] If a name is given for a worksheet that does not exist (e.g., for misspellings), no error results; instead, %exportFromXL simply ignores it. Duplicate-named worksheets are exported only once.
- skipsheets: The names of the worksheets of the Excel input which will *not* be outputted into CSV or TXT, again separated by colons. This is designed for situations where it is simply easier to give the sheets that will *not* be exported. If entries are given for both sheets and skipsheets, the skipsheets parameter is ignored. Similar error checking is done for nonexistent and diplicate names as for sheets.
- lang (optional): The language of the Excel application:[7] *en* (English), *da* (Danish), *de* (German), *es* (Spanish), *fi* (Finnish), *fr* (French), *it* (Italian), *nl* (Dutch), *no* (Norwegian), *pt* (Portuguese), *ru* (Russian) or *sv* (Swedish). **The macro will not work if this parameter is set incorrectly.** Other languages are possible – see Derby (2007). Default value: *en* (English) – although it should be changed appropriately as mentioned in the **INSTALLATION** subsection.
- endclose: Indicates whether Excel should be closed after exporting the data and closing the file. This is useful for saving time when the macro is used repeatedly. Default value: *yes*.

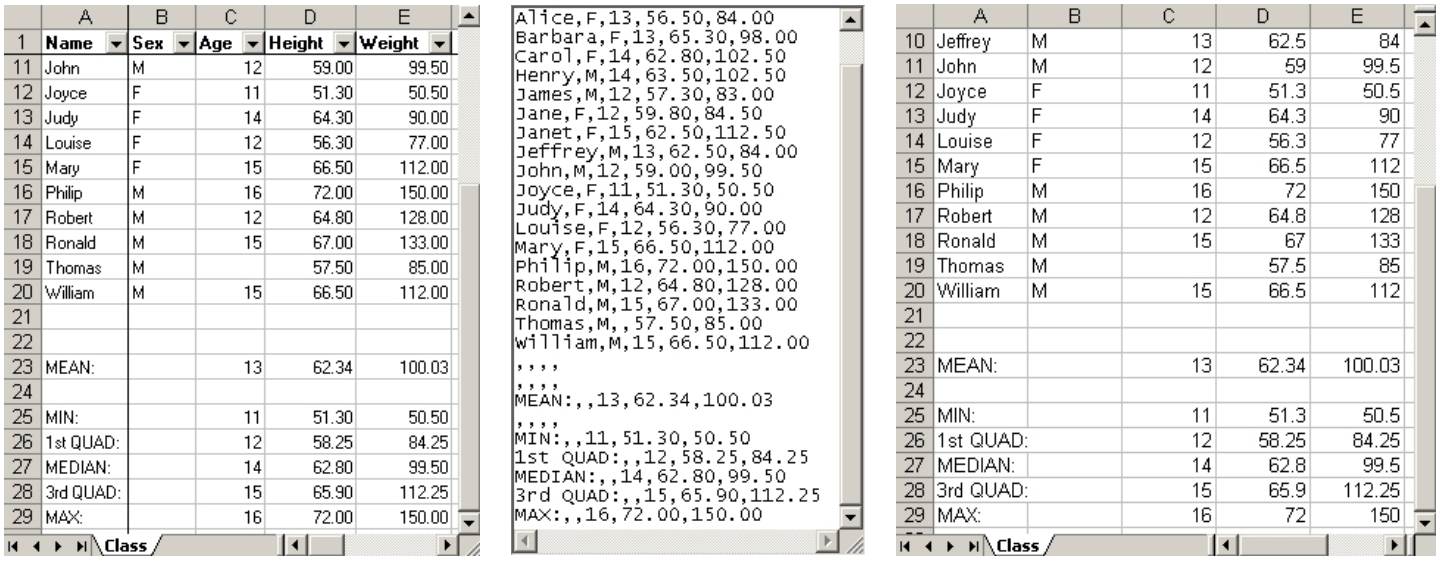
[3] This macro, the code for the following examples, and the user's guide expanded from this paper can all be downloaded from http://exportFromXL.sf.net.

[4] da for Danish, de for German, es for Spanish, fi for Finnish, fr for French, it for Italian, nl for Dutch, no for Norwegian, pt for Portuguese, ru for Russian, or sv for Swedish. **%exportFromXL will not work if this parameter is set incorrectly**.

[5] For the reader who is new to SAS: The %let statement defines a *macro variable* (named *froot*) which stores a collection of characters (*c:\SAS\exportFromXL*). To recall the macro variable later in the code, add an ampersand (&) to the name, as is used above, with &froot. For more details, see Delwiche and Slaughter (2003, p. 200-213). Also, an alternative to this approach would be to save the %exportFromXL macros in a data library and access them via a libname statement.

[6] The colon (:) and the square brackets used in the default ([]) are all characters that are forbidden by Excel to be used in worksheet names. Therefore, using these internal characters in the %exportFromXL component macros will not conflict with any potential worksheet names.

[7] %exportFromXL has only been tested in English and German at this time (8/14/08) – there may be problems with the other languages. See Derby (2008b) or the project website listed under **CONTACT INFORMATION** for updates and details.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Name | Sex | Age | Height | Weight |
| 11 | John | M | 12 | 59.00 | 99.50 |
| 12 | Joyce | F | 11 | 51.30 | 50.50 |
| 13 | Judy | F | 14 | 64.30 | 90.00 |
| 14 | Louise | F | 12 | 56.30 | 77.00 |
| 15 | Mary | F | 15 | 66.50 | 112.00 |
| 16 | Philip | M | 16 | 72.00 | 150.00 |
| 17 | Robert | M | 12 | 64.80 | 128.00 |
| 18 | Ronald | M | 15 | 67.00 | 133.00 |
| 19 | Thomas | M | | 57.50 | 85.00 |
| 20 | William | M | 15 | 66.50 | 112.00 |
| 21 | | | | | |
| 22 | | | | | |
| 23 | MEAN: | | 13 | 62.34 | 100.03 |
| 24 | | | | | |
| 25 | MIN: | | 11 | 51.30 | 50.50 |
| 26 | 1st QUAD: | | 12 | 58.25 | 84.25 |
| 27 | MEDIAN: | | 14 | 62.80 | 99.50 |
| 28 | 3rd QUAD: | | 15 | 65.90 | 112.25 |
| 29 | MAX: | | 16 | 72.00 | 150.00 |

Class

(a)

```
Alice,F,13,56.50,84.00
Barbara,F,13,65.30,98.00
Carol,F,14,62.80,102.50
Henry,M,14,63.50,102.50
James,M,12,57.30,83.00
Jane,F,12,59.80,84.50
Janet,F,15,62.50,112.50
Jeffrey,M,13,62.50,84.00
John,M,12,59.00,99.50
Joyce,F,11,51.30,50.50
Judy,F,14,64.30,90.00
Louise,F,12,56.30,77.00
Mary,F,15,66.50,112.00
Philip,M,16,72.00,150.00
Robert,M,12,64.80,128.00
Ronald,M,15,67.00,133.00
Thomas,M,,57.50,85.00
William,M,15,66.50,112.00
,,,,
,,,,
MEAN:,,13,62.34,100.03
,,,,
MIN:,,11,51.30,50.50
1st QUAD:,,12,58.25,84.25
MEDIAN:,,14,62.80,99.50
3rd QUAD:,,15,65.90,112.25
MAX:,,16,72.00,150.00
```

(b)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 10 | Jeffrey | M | 13 | 62.5 | 84 |
| 11 | John | M | 12 | 59 | 99.5 |
| 12 | Joyce | F | 11 | 51.3 | 50.5 |
| 13 | Judy | F | 14 | 64.3 | 90 |
| 14 | Louise | F | 12 | 56.3 | 77 |
| 15 | Mary | F | 15 | 66.5 | 112 |
| 16 | Philip | M | 16 | 72 | 150 |
| 17 | Robert | M | 12 | 64.8 | 128 |
| 18 | Ronald | M | 15 | 67 | 133 |
| 19 | Thomas | M | | 57.5 | 85 |
| 20 | William | M | 15 | 66.5 | 112 |
| 21 | | | | | |
| 22 | | | | | |
| 23 | MEAN: | | 13 | 62.34 | 100.03 |
| 24 | | | | | |
| 25 | MIN: | | 11 | 51.3 | 50.5 |
| 26 | 1st QUAD: | | 12 | 58.25 | 84.25 |
| 27 | MEDIAN: | | 14 | 62.8 | 99.5 |
| 28 | 3rd QUAD: | | 15 | 65.9 | 112.25 |
| 29 | MAX: | | 16 | 72 | 150 |

Class

(c)

Figure 2: The last observations of the Excel file containing data from `work.class` (a modified form of `sashelp.class`), from the last part of Example 1 of Derby (2008a) (a), and its CSV output, read from Notepad (b) and from Excel (c).

## EXAMPLES

### Example 1: Exporting from One Worksheet into SAS via CSV

For our first example, we can simply export the output from the last part of Example 1 of Derby (2008a), as shown in Figure 2(a). In that example, we exported the contents of `work.class` (a slightly modified form of `sashelp.class`) from SAS into an Excel file with one custom-formatted worksheet, *Class*. Now we can simply export that worksheet from Excel into a CSV file, which will later be imported into SAS via a simple `DATA` step.

```
%exportFromXL( inpath=&inroot, inname=Example 1-7, sheets=Class, outpath=&outroot );
```

The output, named `Class.csv` after its worksheet name, is shown (from within Notepad) in Figure 2(b). Note that the summary statistics shown at the bottom of Figure 2(a), which are Excel formulas in that file, are now stored as hard-coded numbers. This is standard when saving an Excel file as a CSV file.

For an easier view of the data, we may wish to view this CSV file from within Excel, as shown in Figure 2(c). For clarification, this is exactly the same data as shown in Figure 2(b), except that we are now opening it from within Excel. Note that there is now no formatting - e.g., no frozen panes, bold fonts, or an auto filter. This is because the CSV file contains no metadata to store formatting information.

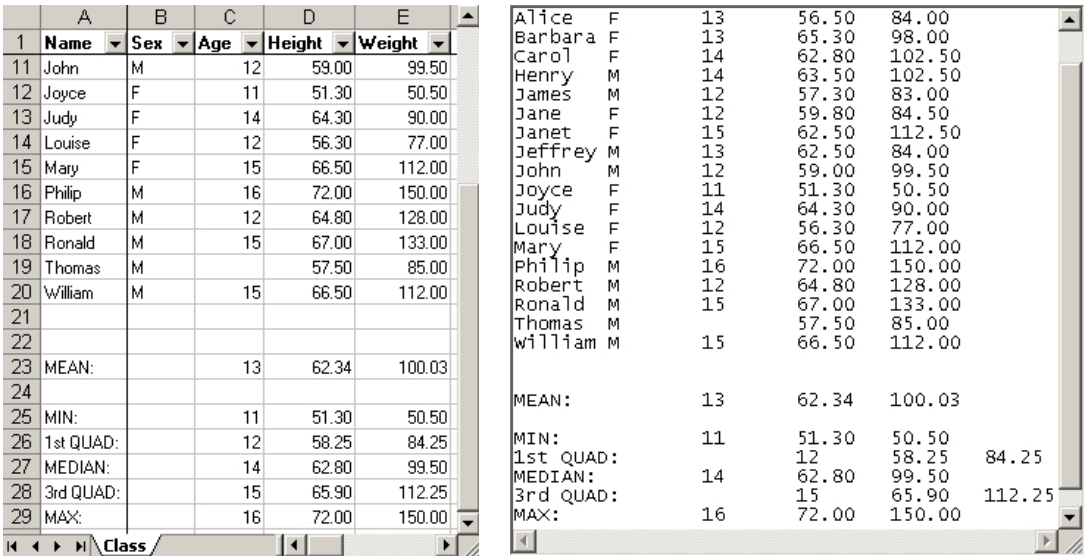
Note that since this Excel file has only one worksheet, we could have called `%exportFromXL` without a `sheets` parameter, which then defaults to exporting all worksheets from the Excel file – which in this case is just *Class*:

```
%exportFromXL( inpath=&inroot, inname=Example 1-7, outpath=&outroot );
```

Now that we have exported *Class* into a CSV file, we can easily read the file into SAS as we would any CSV file:[8]

```
DATA class;
    INFILE "&outroot\Class.csv" DSD MISSOVER FIRSTOBS=2 OBS=20;
    INPUT name :$10. sex :$2. age height weight;
RUN;
```

[8] When `INFILE` is used with a CSV file, SAS assumes that the delimiter value is a comma, so that the option `DLM=','` is not needed.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Name | Sex | Age | Height | Weight |
| 11 | John | M | 12 | 59.00 | 99.50 |
| 12 | Joyce | F | 11 | 51.30 | 50.50 |
| 13 | Judy | F | 14 | 64.30 | 90.00 |
| 14 | Louise | F | 12 | 56.30 | 77.00 |
| 15 | Mary | F | 15 | 66.50 | 112.00 |
| 16 | Philip | M | 16 | 72.00 | 150.00 |
| 17 | Robert | M | 12 | 64.80 | 128.00 |
| 18 | Ronald | M | 15 | 67.00 | 133.00 |
| 19 | Thomas | M | | 57.50 | 85.00 |
| 20 | William | M | 15 | 66.50 | 112.00 |
| 21 | | | | | |
| 22 | | | | | |
| 23 | MEAN: | | 13 | 62.34 | 100.03 |
| 24 | | | | | |
| 25 | MIN: | | 11 | 51.30 | 50.50 |
| 26 | 1st QUAD: | | 12 | 58.25 | 84.25 |
| 27 | MEDIAN: | | 14 | 62.80 | 99.50 |
| 28 | 3rd QUAD: | | 15 | 65.90 | 112.25 |
| 29 | MAX: | | 16 | 72.00 | 150.00 |

Class

(a)

```
Alice   F       13      56.50   84.00
Barbara F       13      65.30   98.00
Carol   F       14      62.80   102.50
Henry   M       14      63.50   102.50
James   M       12      57.30   83.00
Jane    F       12      59.80   84.50
Janet   F       15      62.50   112.50
Jeffrey M       13      62.50   84.00
John    M       12      59.00   99.50
Joyce   F       11      51.30   50.50
Judy    F       14      64.30   90.00
Louise  F       12      56.30   77.00
Mary    F       15      66.50   112.00
Philip  M       16      72.00   150.00
Robert  M       12      64.80   128.00
Ronald  M       15      67.00   133.00
Thomas  M               57.50   85.00
William M       15      66.50   112.00


MEAN:           13      62.34   100.03

MIN:            11      51.30   50.50
1st QUAD:               12      58.25   84.25
MEDIAN:         14      62.80   99.50
3rd QUAD:               15      65.90   112.25
MAX:            16      72.00   150.00
```

(b)

Figure 3: The last observations of the Excel file containing data from `work.class` (a modified form of `sashelp.class`), from the last part of Example 1 of Derby (2008a) (a), and its tab-delimited CSV output, read from Notepad (b).

**Example 2: Exporting from One Worksheet into SAS via tab-delimited TXT**

Now we take the the Excel input file from Example 1 and export it into SAS via a tab-delimited TXT file:[9]

```
%exportFromXL( inpath=&inroot, inname=Example 1-7, sheets=Class, outpath=&outroot, filetype=txt,
  delimiter=tab );
```

The output, named `Class.txt` after its worksheet name once again, is shown in Figure 3(b). As in Example 1, the summary statistics shown at the bottom of Figure 2(a), which are Excel formulas in that file, are now stored as hard-coded numbers. This is again standard when saving an Excel file as a TXT file.

If we were to open this TXT file in Excel, it would look exactly as in Figure 2(c). Again, there is no Excel formatting because there is no metadata stored in a TXT file.

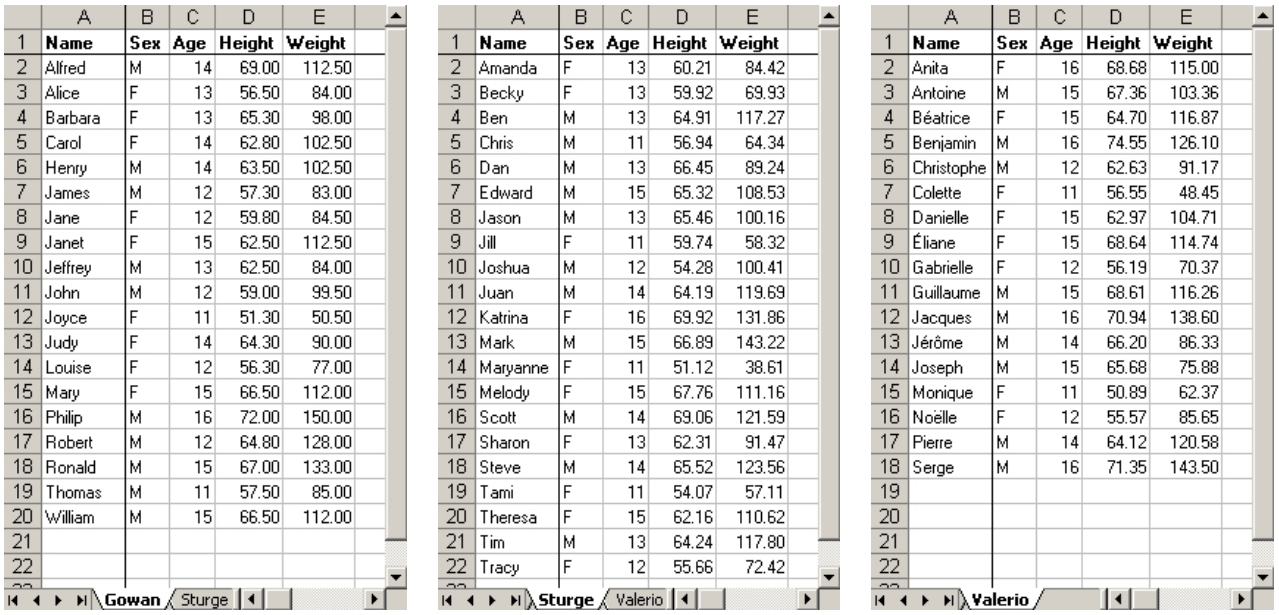
Once again, since this Excel file has only one worksheet, we could have called `%exportFromXL` without a `sheets` parameter, which defaults to exporting all worksheets from the Excel file:

```
%exportFromXL( inpath=&inroot, inname=Example 1-7, outpath=&outroot, filetype=txt, delimiter=tab );
```

Now that we have exported *Class* into a TXT file, we can easily read the file into SAS as we would any TXT file – with the notable addition of `dlm='09'x` to our code from Example 1:

```
DATA class;
    INFILE "&outroot\Class.txt" DSD MISSOVER DLM='09'x FIRSTOBS=2 OBS=20;
    INPUT name :$10. sex :$2. age height weight;
RUN;
```

[9] To export into a *comma*-delimited TXT file, simply set `delimiter=comma` or leave that parameter unspecified (so that it is equal to `comma` by default).



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Name | Sex | Age | Height | Weight |
| 2 | Alfred | M | 14 | 69.00 | 112.50 |
| 3 | Alice | F | 13 | 56.50 | 84.00 |
| 4 | Barbara | F | 13 | 65.30 | 98.00 |
| 5 | Carol | F | 14 | 62.80 | 102.50 |
| 6 | Henry | M | 14 | 63.50 | 102.50 |
| 7 | James | M | 12 | 57.30 | 83.00 |
| 8 | Jane | F | 12 | 59.80 | 84.50 |
| 9 | Janet | F | 15 | 62.50 | 112.50 |
| 10 | Jeffrey | M | 13 | 62.50 | 84.00 |
| 11 | John | M | 12 | 59.00 | 99.50 |
| 12 | Joyce | F | 11 | 51.30 | 50.50 |
| 13 | Judy | F | 14 | 64.30 | 90.00 |
| 14 | Louise | F | 12 | 56.30 | 77.00 |
| 15 | Mary | F | 15 | 66.50 | 112.00 |
| 16 | Philip | M | 16 | 72.00 | 150.00 |
| 17 | Robert | M | 12 | 64.80 | 128.00 |
| 18 | Ronald | M | 15 | 67.00 | 133.00 |
| 19 | Thomas | M | 11 | 57.50 | 85.00 |
| 20 | William | M | 15 | 66.50 | 112.00 |
| 21 | | | | | |
| 22 | | | | | |

Gowan | Sturge

(a)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Name | Sex | Age | Height | Weight |
| 2 | Amanda | F | 13 | 60.21 | 84.42 |
| 3 | Becky | F | 13 | 59.92 | 69.93 |
| 4 | Ben | M | 13 | 64.91 | 117.27 |
| 5 | Chris | M | 11 | 56.94 | 64.34 |
| 6 | Dan | M | 13 | 66.45 | 89.24 |
| 7 | Edward | M | 15 | 65.32 | 108.53 |
| 8 | Jason | M | 13 | 65.46 | 100.16 |
| 9 | Jill | F | 11 | 59.74 | 58.32 |
| 10 | Joshua | M | 12 | 54.28 | 100.41 |
| 11 | Juan | M | 14 | 64.19 | 119.69 |
| 12 | Katrina | F | 16 | 69.92 | 131.86 |
| 13 | Mark | M | 15 | 66.89 | 143.22 |
| 14 | Maryanne | F | 11 | 51.12 | 38.61 |
| 15 | Melody | F | 15 | 67.76 | 111.16 |
| 16 | Scott | M | 14 | 69.06 | 121.59 |
| 17 | Sharon | F | 13 | 62.31 | 91.47 |
| 18 | Steve | M | 14 | 65.52 | 123.56 |
| 19 | Tami | F | 11 | 54.07 | 57.11 |
| 20 | Theresa | F | 15 | 62.16 | 110.62 |
| 21 | Tim | M | 13 | 64.24 | 117.80 |
| 22 | Tracy | F | 12 | 55.66 | 72.42 |

Sturge | Valerio

(b)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Name | Sex | Age | Height | Weight |
| 2 | Anita | F | 16 | 68.68 | 115.00 |
| 3 | Antoine | M | 15 | 67.36 | 103.36 |
| 4 | Béatrice | F | 15 | 64.70 | 116.87 |
| 5 | Benjamin | M | 16 | 74.55 | 126.10 |
| 6 | Christophe | M | 12 | 62.63 | 91.17 |
| 7 | Colette | F | 11 | 56.55 | 48.45 |
| 8 | Danielle | F | 15 | 62.97 | 104.71 |
| 9 | Éliane | F | 15 | 68.64 | 114.74 |
| 10 | Gabrielle | F | 12 | 56.19 | 70.37 |
| 11 | Guillaume | M | 15 | 68.61 | 116.26 |
| 12 | Jacques | M | 16 | 70.94 | 138.60 |
| 13 | Jérôme | M | 14 | 66.20 | 86.33 |
| 14 | Joseph | M | 15 | 65.68 | 75.88 |
| 15 | Monique | F | 11 | 50.89 | 62.37 |
| 16 | Noëlle | F | 12 | 55.57 | 85.65 |
| 17 | Pierre | M | 14 | 64.12 | 120.58 |
| 18 | Serge | M | 16 | 71.35 | 143.50 |
| 19 | | | | | |
| 20 | | | | | |
| 21 | | | | | |
| 22 | | | | | |

Valerio

(c)

Figure 4: The last observations of the Excel file containing internal data sets from Example 5 of Derby (2008a), displaying students in classes of intructors Gowan (a), Sturge (b) and Valerio (c).

**Example 3: Exporting from Many Worksheets into SAS via CSV/TXT**

The past two examples are actually trivial – DDE merely alleviates the need for saving one Excel worksheet as a CSV or TXT file, which can easily be done manually in less than a minute. The real need for a DDE macro like `%exportFromXL` is not for exporting *one* Excel worksheet, but rather *many* – such as, say, 50. For this example, we will merely export three worksheets, but the logic can be extended to any number of worksheets.

For this example, we turn to an internal data set from Example 5 of Derby (2008a). Figure 4 shows data for three classes - those of instructors Gowan, Sturge and Valerio. We would like to export these into a CSV file, and then into SAS. The process is similar if we choose the intermediate file to be TXT rather than CSV.

Certainly we could export them one by one as follows:

```
%exportFromXL( inpath=&inroot, inname=Example 3, sheets=Gowan, outpath=&outroot, endclose=no );
%exportFromXL( inpath=&inroot, inname=Example 3, sheets=Sturge, outpath=&outroot, endclose=no );
%exportFromXL( inpath=&inroot, inname=Example 3, sheets=Valerio, outpath=&outroot );
```

The `endclose=no` parameter states that, for speed, we would rather not close Excel at the end of each export, since we'll have to open it again at the next step. However, we can export all three of the worksheets in one step:

```
%exportFromXL( inpath=&inroot, inname=Example 3, sheets=Gowan:Sturge:Valerio, outpath=&outroot );
```

Note that, unlike for the `exportToXL` macro of Derby (2008a), we now separate the names of the worksheets by a colon (`:`).[10] This is done to allow us to easily include worksheet names that include a space, comma, or other elements of punctuation. Furthermore, since our Excel file only has these three worksheets, we can leave out the `sheets` parameter entirely, and the macro will simply export all worksheets:

```
%exportFromXL( inpath=&inroot, inname=Example 3, outpath=&outroot );
```

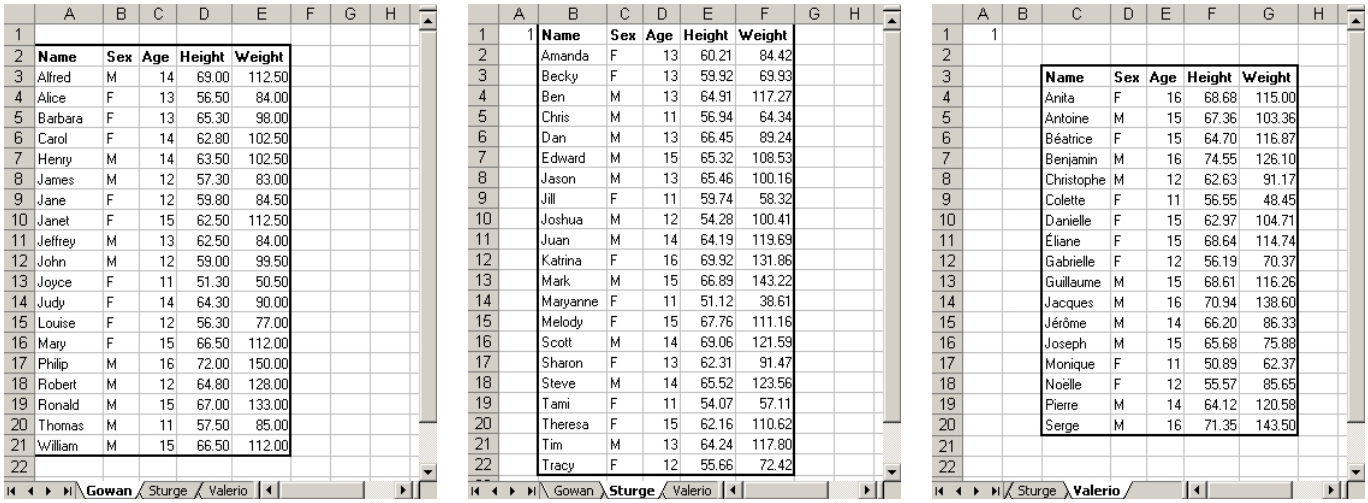
For the last step of exporting the CSV data into SAS, now that we have three data sets of the same structure to import rather than one, it is best to use a clever SAS macro:

```
%MACRO makeSASData( teacher );

  DATA &teacher;
    INFILE "&outroot\%UPCASE( %SUBSTR( &teacher, 1, 1 ) )%SUBSTR( &teacher, 2, %length(&teacher)-1 ).csv"
      DSD MISSOVER FIRSTOBS=2;
    INPUT name :$10. sex :$2. age height weight;
  RUN;

%MEND makeSASData;
```

[10] A colon is forbidden by Excel to be used in the name of a worksheet. Therefore, there will be no possibility of conflicting with a worksheet name.



| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 3 | Alfred | M | 14 | 69.00 | 112.50 |
| 4 | Alice | F | 13 | 56.50 | 84.00 |
| 5 | Barbara | F | 13 | 65.30 | 98.00 |
| 6 | Carol | F | 14 | 62.80 | 102.50 |
| 7 | Henry | M | 14 | 63.50 | 102.50 |
| 8 | James | M | 12 | 57.30 | 83.00 |
| 9 | Jane | F | 12 | 59.80 | 84.50 |
| 10 | Janet | F | 15 | 62.50 | 112.50 |
| 11 | Jeffrey | M | 13 | 62.50 | 84.00 |
| 12 | John | M | 12 | 59.00 | 99.50 |
| 13 | Joyce | F | 11 | 51.30 | 50.50 |
| 14 | Judy | F | 14 | 64.30 | 90.00 |
| 15 | Louise | F | 12 | 56.30 | 77.00 |
| 16 | Mary | F | 15 | 66.50 | 112.00 |
| 17 | Philip | M | 16 | 72.00 | 150.00 |
| 18 | Robert | M | 12 | 64.80 | 128.00 |
| 19 | Ronald | M | 15 | 67.00 | 133.00 |
| 20 | Thomas | M | 11 | 57.50 | 85.00 |
| 21 | William | M | 15 | 66.50 | 112.00 |

Gowan | Sturge | Valerio

(a)

| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 2 | Amanda | F | 13 | 60.21 | 84.42 |
| 3 | Becky | F | 13 | 59.92 | 69.93 |
| 4 | Ben | M | 13 | 64.91 | 117.27 |
| 5 | Chris | M | 11 | 56.94 | 64.34 |
| 6 | Dan | M | 13 | 66.45 | 89.24 |
| 7 | Edward | M | 15 | 65.32 | 108.53 |
| 8 | Jason | M | 13 | 65.46 | 100.16 |
| 9 | Jill | F | 11 | 59.74 | 58.32 |
| 10 | Joshua | M | 12 | 54.28 | 100.41 |
| 11 | Juan | M | 14 | 64.19 | 119.69 |
| 12 | Katrina | F | 16 | 69.92 | 131.86 |
| 13 | Mark | M | 15 | 66.89 | 143.22 |
| 14 | Maryanne | F | 11 | 51.12 | 38.61 |
| 15 | Melody | F | 15 | 67.76 | 111.16 |
| 16 | Scott | M | 14 | 69.06 | 121.59 |
| 17 | Sharon | F | 13 | 62.31 | 91.47 |
| 18 | Steve | M | 14 | 65.52 | 123.56 |
| 19 | Tami | F | 11 | 54.07 | 57.11 |
| 20 | Theresa | F | 15 | 62.16 | 110.62 |
| 21 | Tim | M | 13 | 64.24 | 117.80 |
| 22 | Tracy | F | 12 | 55.66 | 72.42 |

Gowan | Sturge | Valerio

(b)

| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 4 | Anita | F | 16 | 68.68 | 115.00 |
| 5 | Antoine | M | 15 | 67.36 | 103.36 |
| 6 | Béatrice | F | 15 | 64.70 | 116.87 |
| 7 | Benjamin | M | 16 | 74.55 | 126.10 |
| 8 | Christophe | M | 12 | 62.63 | 91.17 |
| 9 | Colette | F | 11 | 56.55 | 48.45 |
| 10 | Danielle | F | 15 | 62.97 | 104.71 |
| 11 | Éliane | F | 15 | 68.64 | 114.74 |
| 12 | Gabrielle | F | 12 | 56.19 | 70.37 |
| 13 | Guillaume | M | 15 | 68.61 | 116.26 |
| 14 | Jacques | M | 16 | 70.94 | 138.60 |
| 15 | Jérôme | M | 14 | 66.20 | 86.33 |
| 16 | Joseph | M | 15 | 65.68 | 75.88 |
| 17 | Monique | F | 11 | 50.89 | 62.37 |
| 18 | Noëlle | F | 12 | 55.57 | 85.65 |
| 19 | Pierre | M | 14 | 64.12 | 120.58 |
| 20 | Serge | M | 16 | 71.35 | 143.50 |

Sturge | Valerio

(c)

Figure 5: The last observations of the Excel file containing internal data sets from Example 5 of Derby (2008a), displaying students in classes of intructors Gowan (a), Sturge (b) and Valerio (c), with variations in the data structure.

Thus, the data can be exported via the following macro calls:

```
%makeSASData( gowan );
%makeSASData( sturge );
%makeSASData( valerio );
```

Or, to be even more clever,

```
%LET teachers = gowan sturge valerio;

%MACRO makeAllSASData( teachers );

  %LOCAL nullid i;
  %LET nullid = ;
  %LET i = 1;
  %DO %UNTIL( %SCAN( &teachers, &i ) = &nullid );
    %makeSASData( %scan( &teachers, &i ) );
    %LET i = %eval( &i + 1 );
    %END;

%MEND makeAllSASData;

%makeAllSASData;
```

While it may seem trivial to do this for 3 worksheets, it can quickly become nontrivial in the case of 50 or more of them.

**Example 4: Exporting from Many Worksheets with Structure Variations into SAS via CSV/TXT**

Now suppose we have a situation just like Example 3, except that the worksheets do not have exactly the same structure. For instance, Figure 5 shows an Excel file with the same data as in 4, except that the Excel file is formatted a bit differently and, more importantly, the data is no longer in exactly the same range. Precisely, Gowan's class starts at row 2, column 1; Sturge's class starts at row 1, column 2; and Valerio's class starts at row 3, column 3. How will this change things?

Happily, we can still account for these variations without giving up our macros. First of all, the first step – exporting the data from Excel into CSV – is still the same:

```
%exportFromXL( inpath=&inroot, inname=Example 4, outpath=&outroot );
```

Note that when exporting data from Excel into a CSV or TXT file, the row and column numbers of the data remain the same. Thus, in this example, Gowan's class data still starts at row 2, column 1; etc.[11] We can work around this with the use of the macro language. One example is the following:

[11] This is only because of the `1` entry in cell A1. If that cell were blank, exporting to CSV/TXT would effectively delete the first columns without data in them. Anyone who exports Excel data to CSV/TXT should remember this fact.

```
%MACRO makeSASData( teacher );

  %LOCAL cell1row cell1col j;

  %IF &teacher = gowan %THEN %DO;
      %LET cell1row = 2;
  %LET cell1col = 1;
      %END;
    %ELSE %IF &teacher = sturge %THEN %DO;
      %LET cell1row = 1;
      %LET cell1col = 2;
      %END;
    %ELSE %IF &teacher = valerio %THEN %DO;
      %LET cell1row = 3;
      %LET cell1col = 3;
      %END;

  DATA &teacher;
    INFILE "&outroot\%UPCASE( %SUBSTR( &teacher, 1, 1 ) )%SUBSTR( &teacher, 2, %LENGTH( &teacher ) - 1 ).csv"
      DSD MISSOVER FIRSTOBS=%EVAL( &cell1row + 1 );
    INPUT %IF &cell1col > 1 %THEN %DO j=1 %TO %EVAL( &cell1col - 1 ); input&j %END;
      name :$10. sex :$2. age height weight;
  RUN;

%MEND makeSASData;
```

Using this, we can then proceed exactly the same way as for Example 3:

```
%LET teachers = gowan sturge valerio;

%MACRO makeAllSASData;

  %LOCAL nullid i;
  %LET nullid = ;
  %LET i = 1;
  %DO %UNTIL( %SCAN( &teachers, &i ) = &nullid );
    %makeSASData( %scan( &teachers, &i ) );
    %LET i = %eval( &i + 1 );
    %END;

%MEND makeAllSASData;

%makeAllSASData;
```

While this is clearly a contrived example, there are very real situations where we can find variations in the data structure.

## LIMITATIONS

While `%exportFromXL` can be very promising for a variety of situations, there are a few limitations:

- It cannot (yet) export hidden worksheets.
- It cannot (yet) sidetrack pop-up security messages that either asks to enable or disable an embedded Excel macro or indicates that such a macro has been disabled. Currently this does not cause `%exportFromXL` to crash - it just pauses the process until an option is clicked by the user, which violates our *Automation* criterion and can dramatically slow down program execution if several Excel files are to be opened and exported into CSV or TXT.
- It only works on PC SAS – and most notably, not on any version of SAS running off of a server, like Enterprise Guide. This is the standard limitation of any method that uses DDE. However, it appears that that there *is* a way to use this (or any DDE macro) from Enterprise Guide if PC SAS is also installed on the (PC) machine in question. This will be investigated in the future – check the project website listed under **CONTACT INFORMATION** for updates and details.

In the above, the qualifier *yet* indicates that the author does not yet know if `%exportFromXL` can be modified to accomodate these limitations. There will be further development work on this in the future. Any user who encounters further limitations (other than those of DDE itself) is encouraged to contact the author or post a message on the project website listed under **CONTACT INFORMATION**.

## CONCLUSIONS

`%exportFromXL` is a SAS macro that, when used on PC SAS with a `DATA` step, can provide an optimal solution for exporting data from Excel into SAS when the criteria are data precision, flexibility, and automation. It performs the first of a two-part

process of first exporting the Excel worksheets into CSV or TXT files, and then importing them into SAS, as illustrated in Figure 1.

Essentially `%exportFromXL` does the same function as an Excel user who clicks on ***File*** → ***Save As...*** and then chooses either the `.csv` or tab-delimited `.txt` option. While this may seem trivial, the lack of automation in the user-directed approach can be a serious disadvantage when confronted with Excel files with a large number of worksheets. Indeed, such situations are not difficult to find in many business settings.

This is just an introduction to `%exportFromXL` – for more information, see Derby (2008b) or the project website listed under **CONTACT INFORMATION**.

## REFERENCES


Delwiche, L. D. and Slaughter, S. J. (2003), *The Little SAS Book*, third edn, SAS Institute, Inc., Cary, NC.

Derby, N. (2007), User's guide to `%exportToXL`, version 1.0.
`http://exportToXL.sf.net/docs/exporttoxlv1.0-ug-cur.pdf`

Derby, N. (2008a), Revisiting DDE: An updated macro for exporting SAS data into custom-formatted Excel spreadsheets, *Proceedings of the 2008 SAS Global Forum*, paper 259-2008.
`http://www2.sas.com/proceedings/forum2008/259-2008.pdf`

Derby, N. (2008b), User's guide to `%exportFromXL`, version 1.0.
`http://exportFromXL.sf.net/docs/exportfromxlv1.0-ug-cur.pdf`

Qi, H. (2004), A practical approach to transferring data from Microsoft Excel to SAS in pharmaceutical research, *Proceedings of the Twenty-Ninth SAS Users Group International Conference*, paper 082-29.
`http://www2.sas.com/proceedings/sugi29/082-29.pdf`

Rashleigh-Berry, R. (2008), `xl2sas` macro.
`http://www.datasavantconsulting.com/roland/xl2sas.sas`

Simonic, A. (2008), WinEdt software.
`http://winedt.com`

Sun, H. and Wong, C. (2005), A macro for importing multiple Excel worksheets into SAS data sets, *Proceedings of the Thirtieth SAS Users Group International Conference*, paper 040-30.
`http://www2.sas.com/proceedings/sugi30/040-30.pdf`

Van Campen, J. (2007), How to use a system of SAS macros for creating Excel worksheets, *Proceedings of the 2007 Western Users of SAS Software Conference*.
`http://www.wuss.org//proceedings07/HandsOnWorkshops/HOW_VanCampen_SASMacrosExcel.pdf`

Vyverman, K. (2000), Using dynamic data exchange to pour SAS data into Microsoft Excel, *Proceedings of the Eighteenth SAS European Users Group International Conference*.
`http://www.sas-consultant.com/professional/SEUGI18-Using-DDE-to-Pour-S.pdf`

Vyverman, K. (2001), Using dynamic data exchange to export your SAS data to MS Excel - Against all ODS, Part I, *Proceedings of the Twenty-Sixth SAS Users Group International Conference*, paper 011-26.
`http://www2.sas.com/proceedings/sugi26/p011-26.pdf`

Vyverman, K. (2003), Fancy MS Word reports made easy: Harnessing the power of Dynamic Data Exchange - Against all ODS, Part II, *Proceedings of the Twenty-Eighth SAS Users Group International Conference*, paper 016-28.
`http://www2.sas.com/proceedings/sugi28/016-28.pdf`

Vyverman, K. (2005), A matter of presentation: Generating PowerPoint slides from Base SAS using Dynamic Data Exchange, *Proceedings of the Thirtieth SAS Users Group International Conference*, paper 045-30.
`http://www2.sas.com/proceedings/sugi30/045-30.pdf`

Watts, P. (2004), Highlighting inconsistent record entries in Excel: Possible with SAS ODS, optimized in Microsoft DDE, *Proceedings of the Seventeenth Northeast SAS Users Group Conference*.
`http://www.nesug.info/Proceedings/nesug04/io/io01.pdf`

Watts, P. (2005), Using single-purpose SAS macros to format Excel spreadsheets with DDE, *Proceedings of the Thirtieth SAS Users Group International Conference*, paper 089-30.
`http://www2.sas.com/proceedings/sugi30/089-30.pdf`

## ACKNOWLEDGMENTS

I am deeply indebted to Koen Vyverman and Perry Watts for their earlier works on this subject – in particular, for Vyverman's work on `%sastoxl`, which is the basis for `%exportToXL`, which in turn is the basis for `%exportFromXL`. I merely filled in the details to their big ideas.

Furthermore, I thank many of the good people at SAS technical support, who kept me going when I got stuck – especially Peter Ruzsa (who made me realize that X4ML commands are language-specific) and Russ Tyndall (who helped me with macro variables).

At SAS I also thank Jim Simon for making my macros that much cleaner.

I thank Ron Fehd for providing me with a LaTeX template for SAS conference papers (used here).

I thank whomever first composed the list of Excel function translations (original source unknown).

Lastly, and most importantly, I thank Charles for his patience and support.

## CONTACT INFORMATION

Comments and questions are valued and encouraged. Contact the author:

Nathaniel Derby
Statis Pro Data Analytics
815 First Ave., Suite 287
Seattle, WA 98104-1404
206-973-2403
`nderby@sprodata.com`
`http://exportFromXL.sf.net`