Paper RF-10-2013

Maintaining Formats when Exporting Data from SAS® into Microsoft® Excel®

Nate Derby, Stakana Analytics, Seattle, WA
Colleen McGahan, BC Cancer Agency, Vancouver, BC

ABSTRACT

Data formats often get lost when exporting from SAS® into Excel® using common procedures such as `PROC EXPORT` or the ExcelXP tagset. In this paper we describe some tricks to retain those formats.

KEYWORDS: SAS, Excel, export, formats.

INTRODUCTION: EXPORTING DATA FROM SAS INTO EXCEL

Many typical ways of exporting data from SAS to Excel, such as `PROC EXPORT` or the ExcelXP tagset, destroy the data formats. To illustrate this, we first create a formatted version of the `sashelp.class` data set, with one student's information set to missing:

```
DATA class;
  SET sashelp.class;
  FORMAT age 3. height weight 6.2;
  IF name = 'Thomas' THEN age = .;
RUN;
```

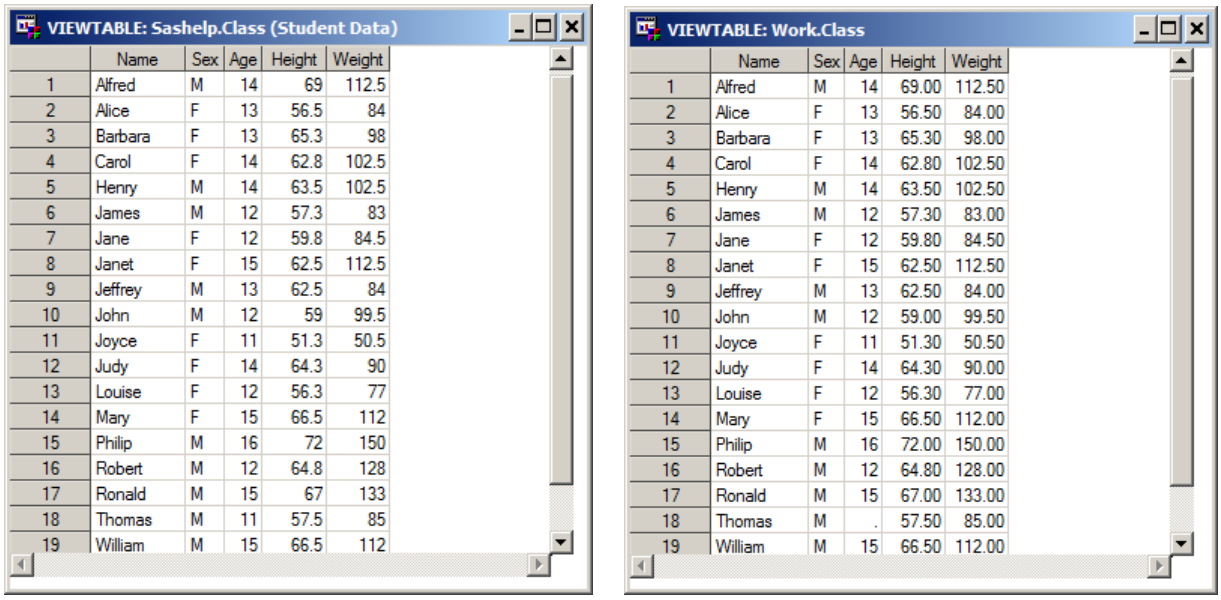
In Figure 1, we can compare the original data set `sashelp.class` (left) with our revised one above, `work.class` (right). We see that in contrast to the original one, the new one has two decimal places for the variables `height` and `weight`. We also have a missing age for Thomas, to illustrate what happens when we export missing data into Excel.

Now that we have our new data set, we export it with `PROC EXPORT` and with the ExcelXP tagset (explained in many papers such as DelGobbo (2006, 2007, 2008, 2009, 2010, 2011, 2012) and Gebhart (2006, 2007a,b, 2008)).

```
PROC EXPORT DATA=class OUTFILE="&outroot\Output from PROC EXPORT.xls";
RUN;

ODS tagsets.ExcelXP FILE="&outroot\Output from ExcelXP.xls";

PROC PRINT DATA=class;
RUN;

ODS tagsets.ExcelXP CLOSE;
```

The ExcelXP tagset is also commonly used with `PROC REPORT`. The syntax would be the following, producing the same output as with `PROC PRINT` above but without the observation numbers (not shown):

```
ODS tagsets.ExcelXP FILE="&outroot\Output from ExcelXP (PROC REPORT).xls";

PROC REPORT DATA=class;
  COLUMN name sex age height weight;
RUN;

ODS tagsets.ExcelXP CLOSE;
```



VIEWTABLE: Sashelp.Class (Student Data)

	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77
14	Mary	F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128
17	Ronald	M	15	67	133
18	Thomas	M	11	57.5	85
19	William	M	15	66.5	112

VIEWTABLE: Work.Class

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

Figure 1: The SAS data sets (left) `sashelp.class` and (right) the formatted version `work.class`.

SAS format	Excel format	Excel format name
`$8.`	`@`	Text
`8.2`	`0.00`	Number, 2 decimal places
`z8.2`	`00000.00`	(none)
`percent8.2`	`0.00%`	Percentage, 2 decimal places
`mmddyy8.`	`mm/dd/yy`	Date, type “03/14/01”
`comma12.2`	`#,##0.00`	Number, 2 decimal places, with comma separator

Table 1: A few SAS formats and their Excel equivalents.

The output for `PROC EXPORT` and for the ExcelXP tagset with `PROC PRINT` are shown in Figure 2. Here we see that the two-decimal-place formats for `height` and `weight` which were in our SAS data sets (right of Figure 1) are now gone. Additionally, the ExcelXP tagset used the period for the missing value (Thomas' age) instead of a blank cell, as is customary for Excel.[1] Both of these are common problems when exporting from SAS into Excel.

This happens because SAS and Excel speak different languages. SAS and Excel formats are coded differently, as shown in Table 1 (from Derby (2008b)). Furthermore, there are some SAS formats without an Excel equivalent and vice versa. Missing data are also coded differently (for numeric variables: the period in SAS, a blank cell in Excel). Many methods for moving data from SAS into Excel move the values but not their formats. We can see this by right clicking on a cell (say, the height for Alfred), then going to *Format Cells...*. As shown in Figure 3, there is no Excel format associated with this cell when using `PROC EXPORT` or the ExcelXP tagset. Therefore, like SAS with unformatted data (one of their few similarities), Excel uses the default form of just displaying decimal places to the last nonzero digit. This is why the heights and weights are shown with either one or no decimal places, just as with the nonformatted data set `sashelp.class` in Figure 1 (left).

To solve this problem, we simply need to translate the format from SAS into Excel. We can do this in a number of ways. This paper describes three of the most common ones.

[1] We actually need a cell with nothing in it (a *null character*), which is different from a blank space inside of a cell, even though they look the same. You can tell the difference by putting the cursor inside of a cell and using the arrow keys to see if there is one or more blank spaces inside the cell. Some Excel operations will work with blank cells but not with a cells containing a blank space. As we will see, some methods in this paper insert a blank space rather than a null character.

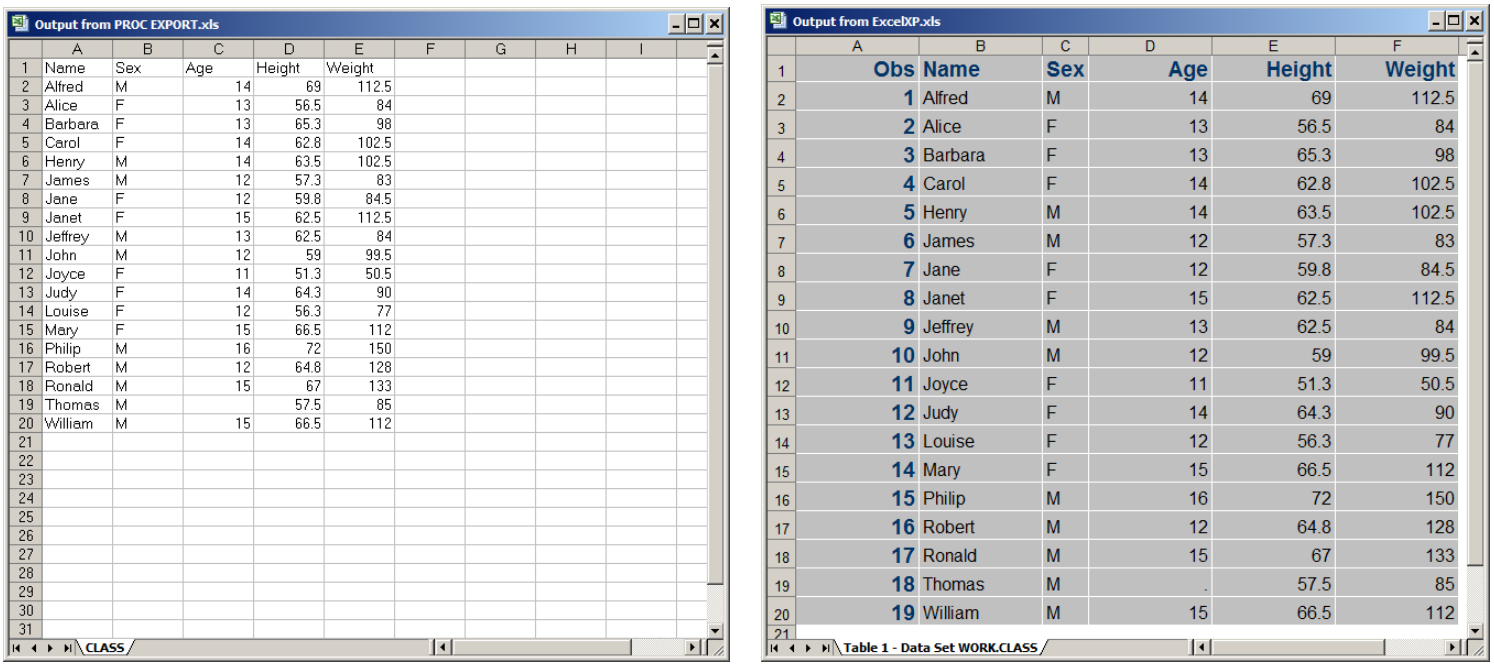


Figure 2: Our formatted SAS data set `work.class` exported into Excel (left) with `PROC EXPORT` and (right) with the ExcelXP tagset.

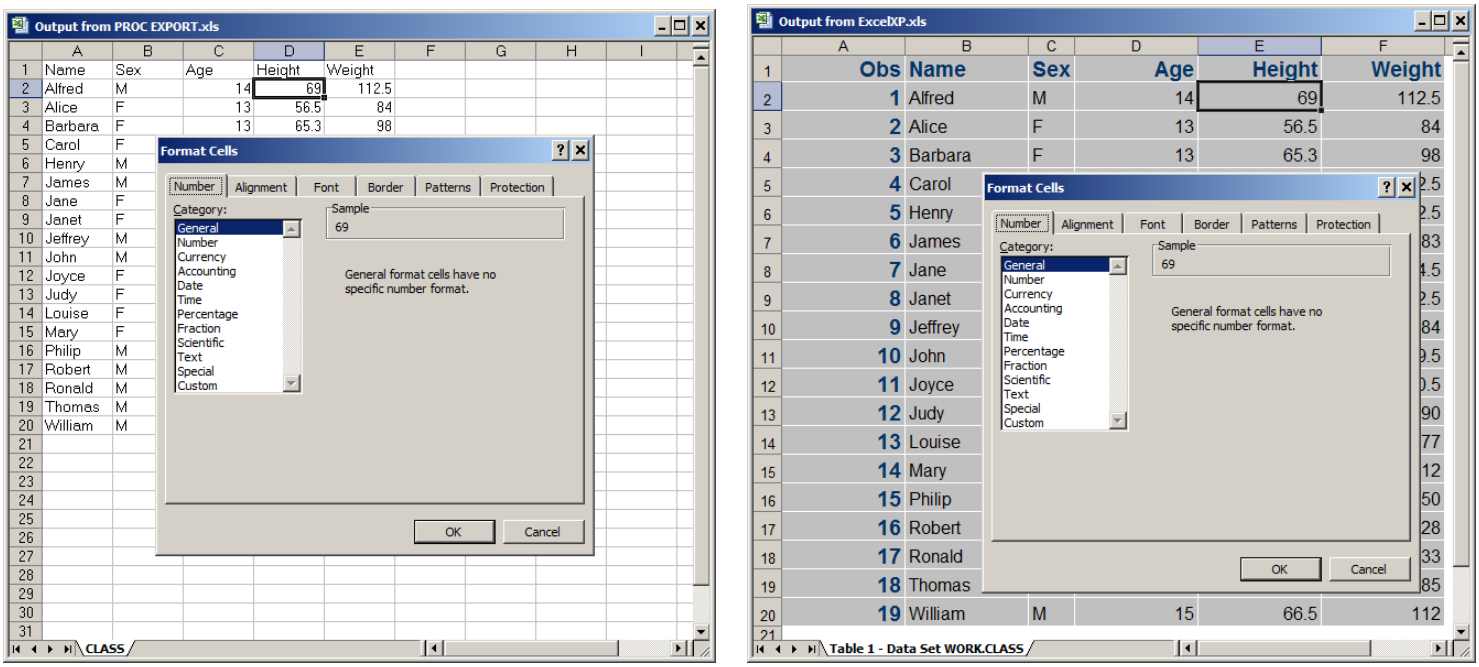


Figure 3: Detail of our formatted SAS data set `work.class` exported into Excel (left) with `PROC EXPORT` and (right) with the ExcelXP tagset.

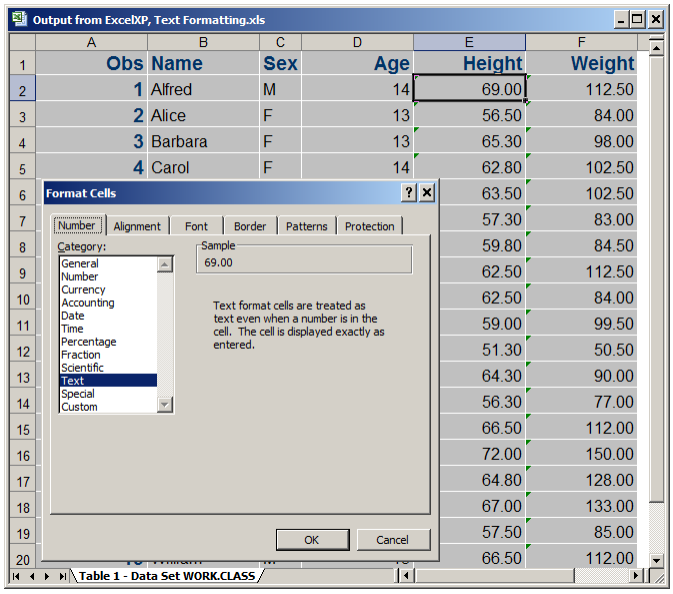


Figure 4: Our formatted data set `work.class` with the ExcelXP tagset formatted as text.

THE EXCELXP TAGSET

Actually, we *can* preserve data formats with the ExcelXP tagset! We just need to tweak it a little.

FORMATTING AS TEXT

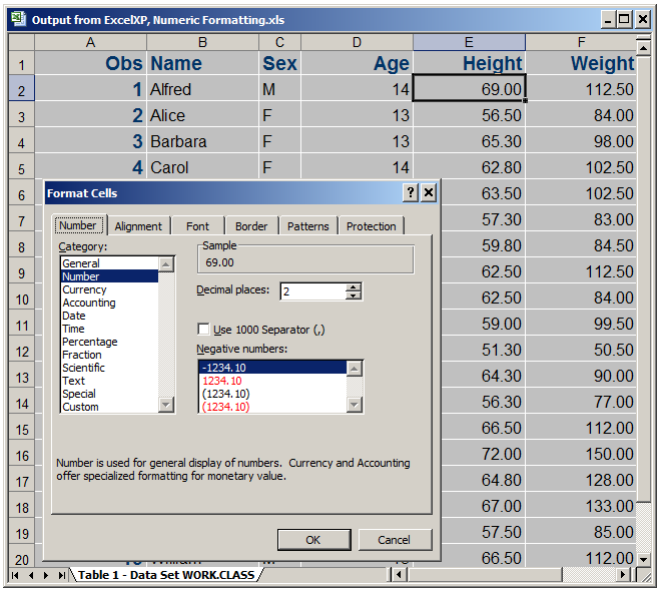
The easiest way to do this is to format everything as text. You can do this with the `TAGGATTR` clause, explained in Gebhart (2010):

```
ODS tagsets.ExcelXP FILE="&outroot\Output from ExcelXP, Text Formatting.xls";

PROC PRINT DATA=class;
  VAR name sex age;
  VAR height weight / STYLE={TAGATTR='format:text'};
RUN;

ODS tagsets.ExcelXP CLOSE;
```

You can also do this with `PROC REPORT` (not shown), which gives about the same output but without the observation numbers.

```
ODS tagsets.ExcelXP FILE="&outroot\Output from ExcelXP, Text Formatting (PROC REPORT).xls";

PROC REPORT DATA=class NOWD;
  COLUMN name sex age height weight;
  DEFINE height / STYLE( column )={TAGATTR='format:text'};
  DEFINE weight / STYLE( column )={TAGATTR='format:text'};
RUN;

ODS tagsets.ExcelXP CLOSE;
```

For `PROC PRINT`, this gives us the output shown in Figure 4. By right clicking on a height cell and looking at the format, we see that it is indeed formatted as a text variable. This is also shown by the small green triangle in the upper left of each cell. This does the job of retaining our data format, but strictly for display purposes. Indeed, since these numbers are treated as text, Excel can't make any calculations (such as sum or average) from these numbers. It would be much better to format them as numbers.



	A	B	C	D	E	F
1	Obs	Name	Sex	Age	Height	Weight
2	1	Alfred	M	14	69.00	112.50
3	2	Alice	F	13	56.50	84.00
4	3	Barbara	F	13	65.30	98.00
5	4	Carol	F	14	62.80	102.50
6					63.50	102.50
7					57.30	83.00
8					59.80	84.50
9					62.50	112.50
10					62.50	84.00
11					59.00	99.50
12					51.30	50.50
13					64.30	90.00
14					56.30	77.00
15					66.50	112.00
16					72.00	150.00
17					64.80	128.00
18					67.00	133.00
19					57.50	85.00
20					66.50	112.00

Figure 5: The correctly formatted output using the ExcelXP tagset with the `TAGATTR` option.

FORMATTING AS A NUMBER

Happily, we can make one small tweak to our code to create a numeric format – in this case, creating two decimal places as in our original data set:

```
ODS tagsets.ExcelXP FILE="&outroot\Output from ExcelXP, Numeric Formatting.xls";

PROC PRINT DATA=class;
  VAR name sex age;
  VAR height weight / style={TAGATTR='format:0.00'};
RUN;

ODS tagsets.ExcelXP CLOSE;
```

In `PROC REPORT`, the code would be the following (output not shown):

```
ODS tagsets.ExcelXP FILE="&outroot\Output from ExcelXP, Numeric Formatting
  (PROC REPORT).xls";

PROC REPORT DATA=class NOWD;
  COLUMN name sex age height weight;
  DEFINE height / STYLE( column )={TAGATTR='format:0.00'};
  DEFINE weight / STYLE( column )={TAGATTR='format:0.00'};
RUN;

ODS tagsets.ExcelXP CLOSE;
```

For `PROC PRINT`, this gives us the output shown in Figure 5. By again right clicking on a height cell and looking at the format, we see that it is indeed formatted as numeric with two decimal places. We have what we wanted! However, it can be tedious to put in a separate `TAGATTR` clause for every numeric variable. This can also create a data set that is larger than it needs to be. We can solve this problem using an *ODS template*.

FORMATTING WITH A TEMPLATE

An *ODS template* is an effective way to create one set of data formats that can be used in several documents. It's described extensively in Haworth et al. (2009), among other books and papers. For a simple illustration, we'll create a style that's the same as the default style, but with the two-decimal format attribute:

```
PROC TEMPLATE;
  DEFINE STYLE styles.mystyle;
    PARENT = styles.default;
    STYLE data_num from data / TAGATTR='format:0.00';
    END;
QUIT;
```

We can then use it with `PROC PRINT` via the following syntax, creating the same output we had before (Figure 5):

```
ODS tagsets.ExcelXP file="&outroot\Output from ExcelXP, Numeric Formatting
  with PROC TEMPLATE.xls" style=mystyle;

PROC PRINT DATA=class;
  VAR name sex age;
  VAR height weight / STYLE( data )=data_num;
RUN;

ODS tagsets.ExcelXP close;
```

With `PROC REPORT`, the syntax would be the following:

```
ODS tagsets.ExcelXP file="&outroot\Output from ExcelXP, Numeric Formatting
  with PROC TEMPLATE (PROC REPORT).xls" style=mystyle;

PROC REPORT DATA=class nowd;
  COLUMN name sex age height weight;
  DEFINE height / STYLE( column )=data_num;
  DEFINE weight / STYLE( column )=data_num;
RUN;

ODS tagsets.ExcelXP close;
```

TREATING MISSING VALUES

Although not shown in Figures 4 or 5, we still have a period for missing values. This can be changed to a blank space by adding

```
OPTIONS MISSING='';
```

before the `ODS` statement and

```
OPTIONS MISSING='.';
```

afterwards to change it back again. However, this produces one blank space rather than a blank cell (i.e., a null character), which can create problems with some Excel operations. As such, this isn't a complete solution.

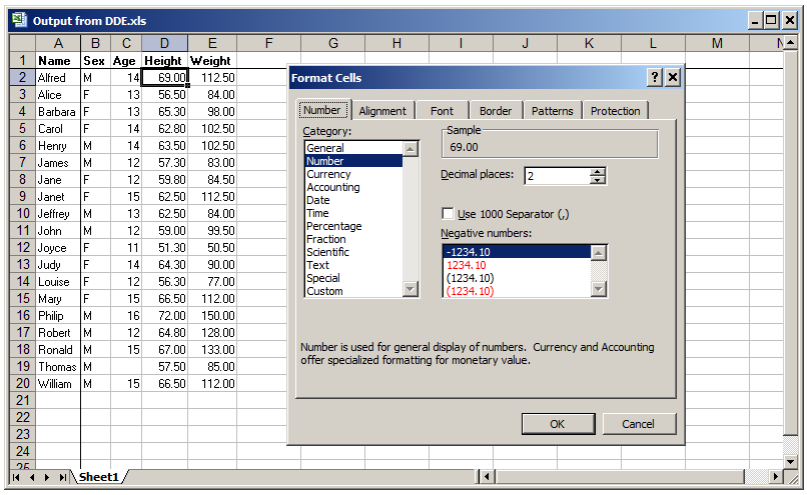


Figure 6: The correctly formatted output using Dynamic Data Exchange via `%exportToXL`.

DYNAMIC DATA EXCHANGE

Dynamic Data Exchange (DDE), as explained in Derby (2008a,b, 2009), Vyverman (2000, 2001, 2002, 2003) and Watts (2004, 2005), involves SAS opening an Excel session and telling Excel what to do. It's only available on PC SAS, as it doesn't work when running on a server (since it needs to access Excel, which isn't on the server!).

However, DDE can be quite frustrating, since you need to program every single step involved. Just opening an Excel session is a multi-step process, as explained in Roper (2000). Fortunately, there's an open-source SAS macro, *exportToXL* (at `exportToXL.net`), which automates this. Due to the way it's written, we must define a macro variable `exroot` that tells SAS where to find it:

```
%let exroot = C:\...\exportToXL;
options sasautos=( "&exroot" ) mautosource;

%exportToXL( dsin=class, savepath=&outroot, savename=Output from DDE );
```

Using this macro produces the output in Figure 6, where we see that the data formats are indeed preserved! But this doesn't happen naturally. There's an entire macro routine happening behind the scenes, `%makeExcelFormats`, which translates the SAS data formats into Excel formats, as shown in Table 1. It works by creating a macro variable of the Excel format and then sending that to Excel. See Derby (2008b) or Vyverman (2003) for details.

The missing age for Thomas is also correctly displayed as a blank cell (rather than a cell with a blank space).[2]

THE `LIBNAME` ENGINE

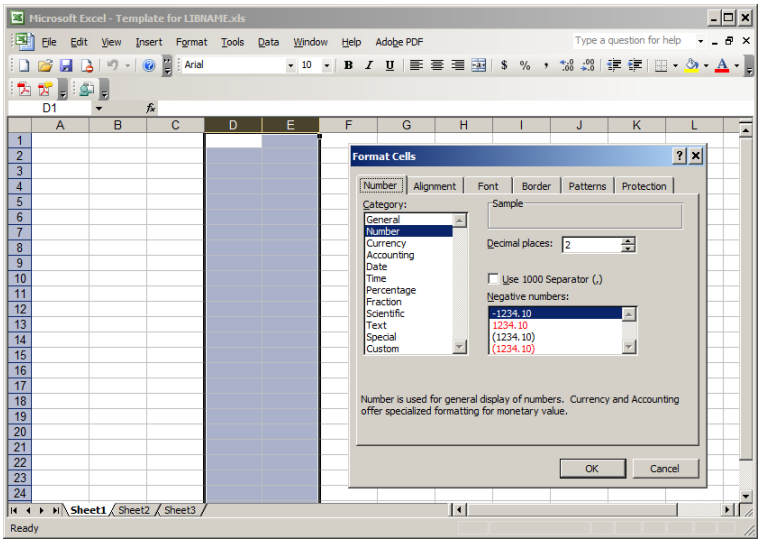
The `LIBNAME` engine allows you to "cheat" by manually formatting the Excel template ahead of time. This is explained in Choate and Martell (2006) and Droogendyk (2008), and requires the SAS/ACCESS to PC Files package. Although this process is simple, it can be tedious if you're moving the data to many different parts of the template, since each part much be formatted ahead of time. Furthermore, be forewarned that there's no way to get rid of the column headers when exporting a data set.

For the preparation, Figure 7 shows the steps: We first (a) pre-format the cells that the numeric variables will be going into (by right clicking, then going to *Format Cells ...*) Then we (b) define and name the range *MyRange* (by going to *Insert* → *Name* → *Define*), then (c) put a border on it.[3]

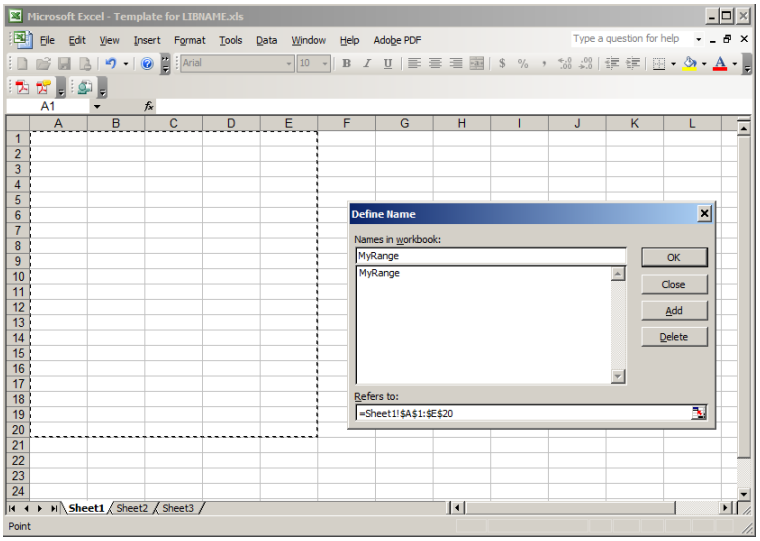
[2] Excel adds a space for each value, so `%exportToXL` simply adds a backspace afterwards.

[3] For some reason, on at least some versions of SAS and Excel, the `LIBNAME` engine deletes the data formats unless some visible format such as a border or cell color is included. This may be a bug, and SAS is looking into it.

(a)



(b)



(c)

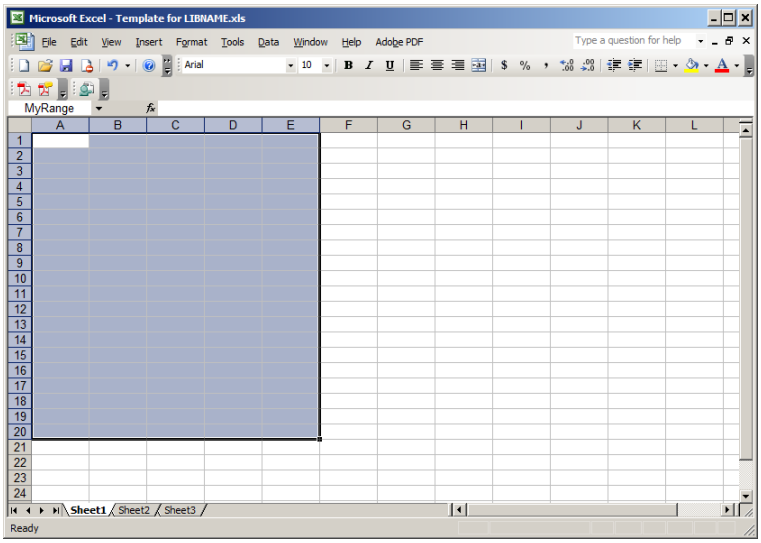
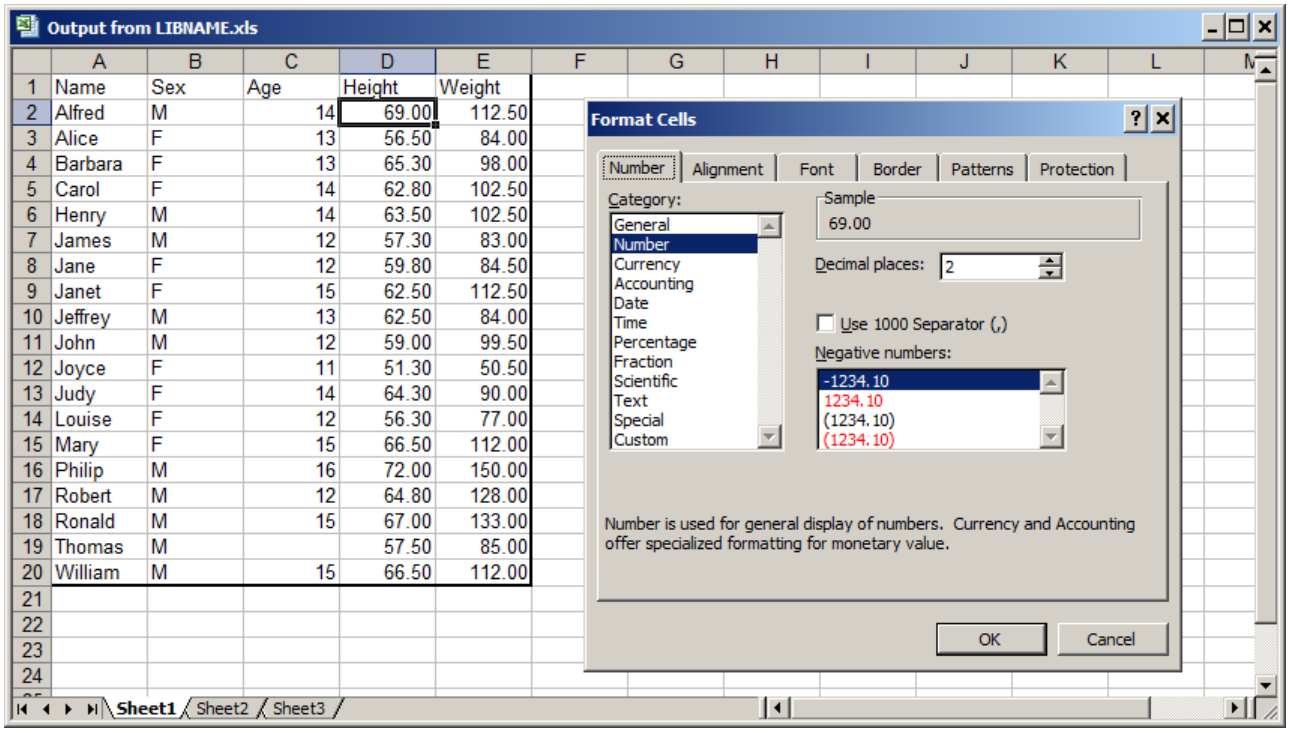


Figure 7: Formatting the Excel template for using the `LIBNAME` engine. We (a) format the columns for two decimal places, (b) define the range, and (c) we're done! But because of a quirk, we must create some kind of visible format – in this case, a border.



	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		57.50	85.00
20	William	M	15	66.50	112.00

Figure 8: The correctly formatted output using the `LIBNAME` engine.

Once this is done, if we rename the template `Output from LIBNAME.xls`, we export the data with the following syntax:

```
LIBNAME workbook PCFILES PATH="&outroot\Output from LIBNAME.xls";

PROC DATASETS LIBRARY=workbook nolist;
  DELETE MyRange;
QUIT;

DATA workbook.MyRange;
  SET class;
RUN;

LIBNAME workbook CLEAR;
```

The `PROC DATASETS` step is required, even though we have no data in the template. The output is shown in Figure 8, where we see that the heights and weights are formatted correctly! Furthermore, the missing age for Thomas is correctly displayed as a blank cell (rather than a cell with a blank space in it).

CONCLUSIONS

Preserving data formats when exporting data from SAS into Excel is a common problem, but fortunately it's common enough that there are solutions. Tweaking the ExcelXP tagset, using Dynamic Data Exchange via the `%exportToXL` macro (which automatically preserves data formats), and pre-formatting the Excel template and then pouring the data into it with the `LIBNAME` engine all do the job well.

In the future, it would be useful to create a macro that automatically detects the variable formats and translates them into the Excel formats for the `TAGATTR` clause of the ExcelXP tagset. Vyverman (2003) and the `%makeExcelFormats` macro of Derby (2008b) explain ideas that could be a good start for doing this.

REFERENCES

Choate, P. and Martell, C. (2006), De-mystifying the SAS LIBNAME engine in Microsoft Excel: A practical guide, *Proceedings of the Thirty-First SAS Users Group International Conference*, paper 024-31.
`http://www2.sas.com/proceedings/sugi31/024-31.pdf`

DelGobbo, V. (2006), Creating and importing multi-sheet Excel workbooks the easy way with SAS, *Proceedings of the Thirty-First SAS Users Group International Conference*, paper 115-31.
`http://www2.sas.com/proceedings/sugi31/115-31.pdf`

DelGobbo, V. (2007), Creating multi-sheet Excel workbooks the easy way with SAS, *Proceedings of the 2007 SAS Global Forum*, paper 229-2007.
`http://www2.sas.com/proceedings/forum2007/229-2007.pdf`

DelGobbo, V. (2008), Tips and tricks for creating multi-sheet Microsoft Excel workbooks the easy way with SAS, *Proceedings of the 2008 SAS Global Forum*, paper 192-2008.
`http://www2.sas.com/proceedings/forum2008/192-2008.pdf`

DelGobbo, V. (2009), More tips and tricks for creating multi-sheet Microsoft Excel workbooks the easy way with SAS, *Proceedings of the 2009 SAS Global Forum*, paper 152-2009.
`http://support.sas.com/resources/papers/proceedings09/152-2009.pdf`

DelGobbo, V. (2010), Traffic lighting your multi-sheet Microsoft Excel workbooks the easy way with SAS, *Proceedings of the 2010 SAS Global Forum*, paper 153-2010.
`http://support.sas.com/resources/papers/proceedings10/153-2010.pdf`

DelGobbo, V. (2011), Creating stylish multi-sheet Microsoft Excel workbooks the easy way with SAS, *Proceedings of the 2011 SAS Global Forum*, paper 170-2011.
`http://support.sas.com/resources/papers/proceedings11/170-2011.pdf`

DelGobbo, V. (2012), An introduction to creating multi-sheet Microsoft Excel workbooks the easy way with SAS, *Proceedings of the 2012 SAS Global Forum*, paper 150-2012.
`http://support.sas.com/resources/papers/proceedings12/150-2012.pdf`

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

Derby, N. (2008b), Revisiting DDE: An updated macro for exporting SAS data into custom-formatted Excel spreadsheets, Part II – Programming details, *Proceedings of the 2008 SAS Global Forum*, paper 260-2008.
`http://www2.sas.com/proceedings/forum2008/260-2008.pdf`

Derby, N. (2009), Creating your own worksheet formats in `exportToXL`, *Proceedings of the 2009 SAS Global Forum*, paper 023-2009.
`http://support.sas.com/resources/papers/proceedings09/023-2009.pdf`

Droogendyk, H. (2008), Customized Excel output using the Excel Libname, *Proceedings of the Fifteenth SouthEast SAS Users Group Conference*.
`http://analytics.ncsu.edu/sesug/2008/SIB-105.pdf`

Gebhart, E. (2006), The beginner's guide to ODS MARKUP: Don't panic!, *Proceedings of the Thirty-First SAS Users Group International Conference*, paper 263-31.
`http://www2.sas.com/proceedings/sugi31/263-31.pdf`

Gebhart, E. (2007a), ODS and Office integration, *Proceedings of the 2007 SAS Global Forum*, paper 227-2007.
`http://www2.sas.com/proceedings/forum2007/227-2007.pdf`

Gebhart, E. (2007b), ODS Markup, tagsets, and styles! Taming ODS styles and tagsets, *Proceedings of the 2007 SAS Global Forum*, paper 225-2007.
`http://www2.sas.com/proceedings/forum2007/225-2007.pdf`

Gebhart, E. (2008), The devil is in the details: Styles, tips, and tricks that make your Microsoft Excel output look great!, *Proceedings of the 2008 SAS Global Forum*, paper 036-2008.
`http://www2.sas.com/proceedings/forum2008/036-2008.pdf`

Gebhart, E. (2010), ODS ExcelXP: Tag Attr it is! using and understanding the TAGATTR= style attribute with the ExcelXP tagset, *Proceedings of the 2010 SAS Global Forum*, paper 031-2010.
http://support.sas.com/resources/papers/proceedings10/031-2010.pdf

Haworth, L., Zender, C. and Burlew, M. (2009), *Output Delivery System (ODS): The Basics and Beyond*, SAS Institute, Inc., Cary, NC.

Roper, C. A. (2000), Intelligently launching Microsoft Excel from SAS, using SCL functions ported to Base SAS, *Proceedings of the Twenty-Fifth SAS Users Group International Conference*, paper 97-25.
http://www2.sas.com/proceedings/sugi25/25/cc/25p097.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. (2002), Creating custom Excel workbooks from Base SAS with Dynamic Data Exchange: A complete walkthrough, *Proceedings of the Twenty-Seventh SAS Users Group International Conference*, paper 190-27.
http://www2.sas.com/proceedings/sugi27/p190-27.pdf

Vyverman, K. (2003), Excel exposed: Using Dynamic Data Exchange to extract metadata from MS Excel workbooks, *Proceedings of the Tenth SouthEast SAS Users Group Conference*.
http://www8.sas.com/scholars/05/PREVIOUS/2001_200.4/2004_MOR/Proceed/_2003/Tutorials/TU15-Vyverman.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

We'd like to thank our supportive employers as well as Vince DelGobbo, Eric Gebhart, Koen Vyverman and Perry Watts, who guided our work in this area.

CONTACT INFORMATION

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

Nate Derby
Stakana Analytics
815 First Ave., Suite 287
Seattle, WA 98104-1404
nderby@stakana.com
http://nderby.org
http://stakana.com

Colleen McGahan
BC Cancer Agency
801 - 686 West Broadway
Vancouver, BC V5Z 1G1
cmcgahan@bccancer.bc.ca

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.