Excellent Ways of Exporting SAS® Data to Excel

Ralph Winters
Alydon Technical Services, Inc.

ABSTRACT

With the addition of the SAS Import/Export Wizard, exporting data to Microsoft® Excel became much easier. However, many alternative solutions exist and this paper will survey the best of them. These include exporting to Excel via the Output Delivery System and using macro code to generate CSV files. Sometimes even simple SAS procedures such as Proc Print can do the trick. This paper is geared toward all SAS users, and describes the benefits of each method as well as some of their pitfalls.

INTRODUCTION

With the introduction of SAS 9, there now exists a plethora of options for exporting data to Microsoft Excel. What used to be a laborious task can now be reduced to several lines of code. Of course, not everyone is currently at SAS 9, nor is everyone using SAS in a Windows environment. In these cases the options are more limited. The choice of which technique you choose ultimately hinges upon the version of SAS you are running, the type of site license you have, the requirements of your application, and of course, personal preference. The goal of this paper is to review some of the best options to perform this seemingly easy task.

The examples in this paper have all been demonstrated using SAS 9.1 and Excel 97 under Windows XP. The sample data sets used are those supplied with SAS.

SAS 9 OPTIONS

Using the LIBNAME Statement

One of the newer ways of writing data to Excel is via the LIBNAME statement. This makes it a snap to simply assign a library reference to an Excel workbook and write directly to its individual sheets. The example below assumes that spreadsheet DEMOA1 will be created from scratch.

```sas
libname myxls "c:\demoA1.xls";

data myxls.houses;
  set sasuser.houses;

data myxls.build;
  set sasuser.build;

data myxls.crime;
  set sasuser.crime;
run;

proc sql;
create table myxls.avg_house_price as
  select style,avg(price) as avg_price from sasuser.houses group by style;
libname myxls clear;
```

The LIBNAME statement assigns a SAS libref to the workbook DEMOA1 on drive C. This can be a brand new workbook or a previously existing workbook. The next three data steps load individual sheets within the workbook corresponding to the sample data sets HOUSES, BUILD and CRIME. Figure 1 shows the exported HOUSES data set. The SQL step shows how you can also create worksheets consisting of summarized data.
In this case we are adding a new sheet called AVG_HOUSE_PRICE that consists of the average price for each house style. Figure 2 displays this result.

Figure 1

<table>
<thead>
<tr>
<th>style</th>
<th>sqfeet</th>
<th>bedrooms</th>
<th>baths</th>
<th>street</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANCH</td>
<td>1250</td>
<td>2</td>
<td></td>
<td>Sheppard Avenue</td>
<td>$64,000.00</td>
</tr>
<tr>
<td>SPLIT</td>
<td>1190</td>
<td>1</td>
<td></td>
<td>Rand Street</td>
<td>$65,850.00</td>
</tr>
<tr>
<td>CONDO</td>
<td>1400</td>
<td>2</td>
<td>1.5</td>
<td>Market Street</td>
<td>$80,050.00</td>
</tr>
<tr>
<td>TWO STORY</td>
<td>1810</td>
<td>4</td>
<td>3</td>
<td>Garris Street</td>
<td>$107,250.00</td>
</tr>
<tr>
<td>RANCH</td>
<td>1500</td>
<td>3</td>
<td></td>
<td>Kemble Avenue</td>
<td>$86,850.00</td>
</tr>
<tr>
<td>SPLIT</td>
<td>1615</td>
<td>4</td>
<td>3</td>
<td>West Drive</td>
<td>$94,450.00</td>
</tr>
<tr>
<td>SPLIT</td>
<td>1305</td>
<td>3</td>
<td>1.5</td>
<td>Graham Avenue</td>
<td>$73,650.00</td>
</tr>
<tr>
<td>CONDO</td>
<td>1390</td>
<td>3</td>
<td>2.5</td>
<td>Hampshire Avenue</td>
<td>$79,350.00</td>
</tr>
<tr>
<td>TWO STORY</td>
<td>1040</td>
<td>2</td>
<td>1</td>
<td>Sanders Road</td>
<td>$55,850.00</td>
</tr>
<tr>
<td>CONDO</td>
<td>2105</td>
<td>4</td>
<td>2.5</td>
<td>Jeans Avenue</td>
<td>$127,150.00</td>
</tr>
<tr>
<td>RANCH</td>
<td>1535</td>
<td>3</td>
<td></td>
<td>State Highway</td>
<td>$89,100.00</td>
</tr>
<tr>
<td>TWO STORY</td>
<td>1240</td>
<td>2</td>
<td>1</td>
<td>Fairbanks Circle</td>
<td>$69,250.00</td>
</tr>
<tr>
<td>RANCH</td>
<td>1720</td>
<td>1</td>
<td>1</td>
<td>Nicholson Drive</td>
<td>$34,550.00</td>
</tr>
<tr>
<td>TWO STORY</td>
<td>1745</td>
<td>4</td>
<td>2.5</td>
<td>Highland Road</td>
<td>$102,950.00</td>
</tr>
<tr>
<td>CONDO</td>
<td>1860</td>
<td>2</td>
<td>1</td>
<td>Arcata Avenue</td>
<td>$110,700.00</td>
</tr>
</tbody>
</table>
One thing that you will discover is that occasionally the original SAS formatting is not preserved. Note that variable AVG_PRICE in Figure 2 is no longer formatted appropriately. Formatting is performed according to the conversion rules listed in Table 1, which will usually give acceptable results. If formatting is critical to your application, you may need to consider other options (DelGobbo SUGI 28).

<table>
<thead>
<tr>
<th>XLS Data Type</th>
<th>SAS Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEXT</td>
<td>$BINARYw.</td>
</tr>
<tr>
<td></td>
<td>$CHARw.</td>
</tr>
<tr>
<td></td>
<td>$HEXw.</td>
</tr>
<tr>
<td></td>
<td>$w.</td>
</tr>
<tr>
<td>NUMBER</td>
<td>w.d</td>
</tr>
<tr>
<td></td>
<td>BESTw.</td>
</tr>
<tr>
<td></td>
<td>BINARYw.</td>
</tr>
<tr>
<td></td>
<td>COMMAw.d</td>
</tr>
<tr>
<td></td>
<td>COMMAXw.</td>
</tr>
<tr>
<td></td>
<td>Ew.</td>
</tr>
<tr>
<td></td>
<td>FRACTw.</td>
</tr>
<tr>
<td>CURRENCY</td>
<td>DOLLARw.d</td>
</tr>
<tr>
<td></td>
<td>DOLLARXw.d</td>
</tr>
<tr>
<td>DATETIME</td>
<td>DATEw.</td>
</tr>
<tr>
<td></td>
<td>DATETIMEw.d</td>
</tr>
<tr>
<td></td>
<td>DDMYYYw.</td>
</tr>
<tr>
<td></td>
<td>HHMMw.d</td>
</tr>
<tr>
<td></td>
<td>JULDAYw.</td>
</tr>
<tr>
<td></td>
<td>JULIANw.</td>
</tr>
<tr>
<td></td>
<td>MMDDYYw.</td>
</tr>
<tr>
<td></td>
<td>MOYyw.</td>
</tr>
<tr>
<td></td>
<td>WEEKDATEw.</td>
</tr>
<tr>
<td></td>
<td>WEEKDATXw.</td>
</tr>
<tr>
<td></td>
<td>WEEKDAYw.</td>
</tr>
<tr>
<td></td>
<td>WORDDATEw.</td>
</tr>
<tr>
<td></td>
<td>WORDDATXw.</td>
</tr>
</tbody>
</table>
SAS dates can be troublesome as well, so make sure you specify a date format in your code:

```sas
data datetest;
  format date_formatted date9.;
  date_formatted="13MAY2004"d;
  date_unformatted="04JUL2004"D;
run;
libname myxls "c:\demodate.xls" ;
data myxls.tab2 ;
  set datetest;
```

In this case Excel will correctly display 05/13/2004 for the variable DATE_FORMATTED in worksheet TAB2 of workbook DEMODATE. However variable DATE_UNFORMATTED will display as 16256, which is the unformatted numeric value for "04JUL2004"D.

There is a variation of this code that can be run to create a spreadsheet from the UNIX operating system, if SAS/ACCESS for PC files has been installed. This technique uses the PCFILES option in the SAS LIBNAME statement:

```sas
libname myxls pcfiles server=D2323 port=8621 path="c:\demo3.xls";
```

**ODS CSV**

The ODS CSV option that was experimental in SAS 8 is production in SAS 9. This feature allows you to specify an ODS CSV destination for a procedure, which then writes its output as a CSV (comma separated value) file.

Here is a simple example using the sample data set SASUSER.CLASS. The CSV output is shown in Figure 3, as displayed by Excel.

```sas
ODS CSV file='myfile.csv';
proc print data=sasuser.class;
run;
ODS CSV close;
```
Let's use the ODS CSV technique on our dates data set to see how EXCEL will handle these values.

```sas
data datetest;
   format date_formatted date9.;
   date_formatted="13MAY2004"d;
   date_unformatted="04JUL2004"D;
run;

ODS CSV file= "c:\demodate.csv";
proc print data=datetest;
run;
ODS CSV close;
run;
```

Here our formatted and unformatted dates will be treated as they were previously, although variable DATE_FORMATTED will be presented as 13-May-04 instead of 05/13/2004.

**SAS 7 & 8 OPTIONS**

In SAS 7 and 8 there are still many excellent ways to export data, including the SAS System Viewer and ODS HTML.
The SAS System Viewer

The SAS Viewer is a free Windows application and a great tool for copying SAS data sets to Excel. It is not even necessary to have SAS installed on your computer, so it is equally useful for programmers and non-programmers alike. To use it you simply start the viewer, open the data set, select your data set, then cut and paste it into Excel. Format your Excel cells as all text in order to preserve the data’s formatting. The viewer will allow you to save the data as a CSV or TXT file and performs simple filtering and sorting of data. This software will also support some cross-platform SAS data sets, so it is a great tool for multiple OS environments.

The following example use the sample data set SASUSER.CLASS.

- Start the SAS System Viewer.
- Select FILE and then OPEN to get to the file selector dialog. Find and open SASUSER.CLASS.
- Enter a CTRL-A followed by a CTRL-C to select and copy the entire table. You can also select portions of the table as needed.

Let’s filter the data set to include only the males:

- Select Edit/Filter from the Viewer drop down menu.
- Type the WHERE clause “Where Sex='M'” and press ENTER. The results are shown in Figure 4.

The next step is to decide which method to choose to export to EXCEL.

- If the table is small enough and/or partial columns or rows have been selected, it is easy enough to SELECT then PASTE the data directly into EXCEL.
- You can also save the entire table as a CSV file using the File/Save dialog, then import it into Excel.

Figure 4

![Filtering SAS data](image-url)
ODS HTML

ODS offers another way to export your data to Excel – ODS HTML. Although any template can be used to illustrate this method, if you are concerned mainly with moving your data, using the MINIMAL style will allow you to import your data into Excel with a minimum of HTML formatting. The results of the code below are displayed in Figure 5.

ODS HTML file='H:\myfile.xls' RS=none style=MINIMAL;

proc print data=sashelp.class noobs;

ODS HTML CLOSE;
run;

Figure 5

![Excel spreadsheet image]

Even though the MINIMAL style incorporates the least HTML output into your spreadsheet, the result is still HTML. One way to completely eliminate to this HTML is to create a new worksheet, select the HTML output, then perform a Paste/Special into the new worksheet, making sure that the “Values Only” button is selected. In the ODS HTML sample code above I have used Proc Print purely for its simplicity, but you need not stop there. Many applications make extensive use of HTML output to produce very sophisticated displays generated by Proc Report and Proc Tabulate.

ODS CSV or ODS HTML?

We have now seen two ODS options that can be used to export SAS data to a spreadsheet -- ODS CSV and ODS HTML. So which one is better? ODS CSV will produce cleaner output, and would be preferable when automating exports to spreadsheets where moving data is the central focus. Use ODS HTML instead when you wish to retain some or all of the HTML formatting. This would usually be for presentation purposes.
SAS 6 OPTIONS

SAS 6.12 first introduced the File Export Wizard, which has been gradually improved upon since its initial release. The purpose of the wizard is to export SAS data sets to a variety of formats that can then be read by most external applications, including Microsoft Excel.

To use the File Export Wizard:

- Choose File Export.
- Specify the library and member (SAS data set) name. For example SASUSER.CLASS.
- Press NEXT.
- Select Tab Delimited format (*.txt).
- Press NEXT.
- Specify the full path where you want to save the file (c:\filename.txt, for example).
- Press FINISH. You will see a message in the log indicating that the file was successfully created.

Tab delimited files were chosen in this instance in order to accommodate embedded commas in the source data. In addition, this is a universal method that can be used across all platforms.

Another way to harness the power of the File Export Wizard is to use the SAS Display Manager command DEXPORT. The steps listed above, for example, can be duplicated with this single command:

DEXPORT SASUSER.CLASS “C:\FILENAME.TXT”

This following example uses the SAS DM command to automatically output the contents of data set SAMPLE to an external file:

```sas
data sample;
  do i=1 to 10;
    output;
  end;
run;

dm "dexport sample 'C:\filename.txt'  replace";
```

OPTIONS PRIOR TO SAS 6.12

SAS versions prior to SAS 6.12 require additional creativity in terms of interfacing SAS with Excel.

Using DDE

DDE is a Microsoft data exchange protocol that works under OS/2 and Windows. One advantage that it enjoys is that it will work with older versions of both SAS and of Excel. The simplest way to use this method is to open Excel with a blank workbook and select the default worksheet SHEET1. Now switch back to SAS in order to define the connection between SAS and Excel. This is done with a FILENAME statement in the form:

FILENAME fileref DDE 'DDE-triplet' <DDE-options>

The DDE triplet is simply the sheet name, row and column area where the data is to be inserted. Let’s assume that you know your data to be contained within 100 rows and 13 columns. In addition, you want to anchor your data in Row 1, Column 1 of the target worksheet. You would use this SAS code:
If you can live with the inconvenience of not exporting label formats, this is a fairly quick way to get data into a spreadsheet. However this will not work for variables which contain embedded blanks. For this case, you need to add the NOTAB option and explicitly write the tab delimiter. Below is an example which outputs all variables in a tab delimited format. The results are displayed in Figure 6.

```sas
filename ddedata DDE 'excel|sheet2!r1c1:r100c13' notab;
data null;
set sasuser.crime;
where staten=": "New";
file ddedata;
put (_all_) ("09"x);
```

Figure 6

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>New Hamp</td>
<td>33</td>
<td>3.2</td>
<td>10.7</td>
<td>23.2</td>
<td>76</td>
<td>1041.7</td>
<td>2343.9</td>
<td>293.4</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>New Jer</td>
<td>34</td>
<td>5.6</td>
<td>21.4</td>
<td>180.4</td>
<td>185.1</td>
<td>1435.8</td>
<td>2774.5</td>
<td>511.5</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>New Mex</td>
<td>35</td>
<td>8.8</td>
<td>39.1</td>
<td>108.6</td>
<td>343.4</td>
<td>1418.7</td>
<td>3008.8</td>
<td>259.5</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>New York</td>
<td>36</td>
<td>10.7</td>
<td>29.4</td>
<td>472.6</td>
<td>319.1</td>
<td>1728</td>
<td>2782</td>
<td>745.8</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After the worksheet is populated, we are left with a blank column A, so we simply delete it.

**Cut and Paste**

Considered primitive by many, this method can actually be the quickest way to export small listings that already exist on flat files or in the output window, or when you can’t change the SAS program that generates the output. This is also a very useful method when your SAS output is created under UNIX and your need to load a Windows spreadsheet. In this case, it is absolutely required that some character be present for all variable values, so it may be necessary to perform some quick editing before pasting the data into Excel.

For our example, we will use the SAS listing displayed in Figure 7.

**Figure 7**

<table>
<thead>
<tr>
<th>Obs</th>
<th>staten</th>
<th>state</th>
<th>murder</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>New Hampshire</td>
<td>33</td>
<td>3.2</td>
</tr>
<tr>
<td>30</td>
<td>New Jersey</td>
<td>34</td>
<td>5.6</td>
</tr>
<tr>
<td>31</td>
<td>New Mexico</td>
<td>35</td>
<td>8.8</td>
</tr>
<tr>
<td>32</td>
<td>New York</td>
<td>36</td>
<td>10.7</td>
</tr>
<tr>
<td>33</td>
<td>North Carolina</td>
<td>37</td>
<td>10.6</td>
</tr>
<tr>
<td>34</td>
<td>North Dakota</td>
<td>38</td>
<td>0.9</td>
</tr>
<tr>
<td>35</td>
<td>Ohio</td>
<td>39</td>
<td>7.8</td>
</tr>
</tbody>
</table>
A general procedure to use when faced with output like this is:

- Select all the data from the output window, cut and paste it into Excel. All the data should end up contained within Column A.
- In Excel, sort the data by Column A, which will aggregate all the paired observation records together, by OBS number.
- Get rid of extraneous lines like titles, page numbers etc.
- Join all paired observations (row 1-2, 3-4, etc.) using the Excel CONCATENATE function. Then use Copy/Paste/Values to copy that column (column G in our example) to another worksheet. See Figure 8.
- Use the ‘Text to Column function’ to parse out the concatenated data line. Use the first data row as the template line and use ‘Fixed Width’ to insert column breaks. Excel will usually make pretty good guesses as to where the lines should be. In this sample, our data contains embedded blanks (e.g. ‘New Jersey’) so we need to be careful before we accept Excel’s default split.
- Sort on one column in order to sort out the joined data line from the paired segment lines. In our example I know that the state name is ending up in column I, so I will sort on that column and then edit out all of extraneous data contained in rows 1-6 and 13-16, as well as extraneous columns A and H. See Figure 9.
- Because of the way we have selected our data, the order of the data fields is now rape, robbery, assault, burglary, larceny, auto, Staten, state and murder. These can now be inserted as labels.

Figure 8
The CSV Macro Approach

A macro that generates a CSV file is very easy to call from existing applications. A robust design, which can accept a variety of user-supplied parameters, can accommodate both a variety of input data set types as well as the several versions of SAS currently in use.

Here is the call for the MAKEFILE macro supplied by SAS Institute.

```sas
%makefile(dataset=sasuser.houses,
   filename="csvmacro2.csv", /* FILEREF or DDNAME of the file */
   dlmr=",",
   quote="no",
   header="no",
   label="no");
```

This call will generate a CSV file from the SASUSER.HOUSES data set. I will not discuss the code behind the MAKEFILE macro at length, although I would like to mention a couple of interesting things about it:

- The macro uses a Proc Contents to generate a list of variables, labels, and formats from the original data set.
• The macro dynamically generates a PUT statement to output the variables based upon the input parameters. The format of the generated PUT statement (using the SAS supplied example) is:

\[
\text{put id +(-1) "", "" name +(-1) "", "" amount +(-1) "", "" date ;}
\]

SUMMARY

In this paper I have tried to show the variety of techniques available for outputting SAS data to Excel. In addition to those discussed here, there are many newly emerging techniques which are now or will soon be available, including XML, the ExcelXP Tagset and Proc SYLK. These I leave to the reader to explore.

REFERENCES


CSV Macros


ACKNOWLEDGEMENTS

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. Microsoft and Microsoft Excel are U.S. registered trademarks of Microsoft Corporation. Other brand and product names are trademarks of their respective companies.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.
Contact the author at:

Ralph Winters
Email: rwinters2@verizon.net