CONVERTING SAS DATA SETS TO OTHER FILE TYPES

Several ways exist to import data from external files into SAS, such as data stored in Excel worksheets.

SAS has an EXPORT wizard from the menus and a procedure called PROC EXPORT that work in an analogous manner to the IMPORT process. Exporting data through them is satisfactory when you do not need to specify how the data will be formatted in the resulting Excel worksheet.

Exporting one SAS dataset to a single worksheet in a new Excel workbook with PROC EXPORT is quite simple. You need to remember that Excel is limited to a maximum size of 256 columns (variables) by 65536 rows (observations). If your dataset is larger in either dimension, you'll need to split it into smaller subsets and merge or append them together.

PROC EXPORT and the Export Wizard are helpful tools for exporting SAS datasets to other programs or files. This procedure is available for converting SAS datasets to comma delimited files and excel files in versions 97, 2000, and 2002 under Windows.

One application of PROC EXPORT is to write SAS datasets to other formats when you have specific analyses, graphics, or functions you wish to perform with other software, or when transferring your data to other users who may not be able to run SAS software.

As with PROC IMPORT, the Export Wizard steps you through the process of writing an Excel workbook, while at the same time (optionally) writing and saving the program statements. To use the Export Wizard:

* With the SAS Program Editor active, select File Export Data
* Select the folder and the specific data set that you wish to export

You will also be given the option of naming a file for saving the program statements. Doing this is recommended to document the conversion procedure. It is simple to rerun the program to update your export program file should you update or correct SAS data sets, rather than step through the wizard again and again.

When you export a SAS dataset to a delimited file the column names will automatically be entered in the first row. It is not possible to omit them directly within PROC EXPORT. But for clear documentation, why would you want to? You would lose the variable names that specify their inherent order in the data file. Without them could cause you some serious problems later on.

Other options for specialized formatting of data or saving only a subset of variables or records are available. See the online help, or the SAS procedures manual for PROC EXPORT.

Converting a SAS Dataset to an External Text File

With a DATA step you can write the contents of a dataset set or specify selected variables to an external text file with your choice of delimiters:

```
FILENAME dat "c:\data\test_0999.txt";

DATA _null_;
SET sashelp.class;
FILE dat;
*PUT (_all_) ('09'x);  * For tab separated;
*PUT (_all_) (',');    * For comma separated;
PUT (_all_) (:);       * For space separated;
RUN;
```

To write a comma-separated-value file (CSV), the Output Delivery System quickly does the job:

```
ODS tagsets.csv body='c:\sas\data\farms.csv' ;
PROC PRINT data=farms NOobs;
TITLE " ";
run;

ODS tagsets.csv close;
```

PROC EXPORT converts a SAS dataset to a text file with several possible delimiters. Assume you have a SAS dataset on your computer called mydat.sas7bdat that is located in the directory c:\sas\data and you want to convert it into a comma separated value text file called mydat.csv. The following statements in PROC EXPORT will convert this SAS data set into a text file called mydat.csv.

```
LIBNAME dat "c:\sas\data\";

PROC EXPORT DATA=dat.mydat
        OUTFILE="c:\sas\data\mydat.csv"
        DBMS=csv REPLACE;
RUN;
```

Convert a SAS Dataset into an Excel Worksheet

The following export procedure places a SAS dataset into an Excel worksheet. This example writes an Excel file called inventory.xls with a worksheet name called 'prices' (observe this name in the lower left corner tab of the Excel worksheet). SAS places the variable names in Row 1 of the Excel worksheet.

```
DATA prices;
        FORMAT product $2. ;
        INPUT @1 product $2. new_price on_hand ;
CARDS;
AA  16.95 5
BB  19.99 6
CC  10.45 8
AA  22.01 9
CC  21.35 1
;

PROC EXPORT DATA=prices
        OUTFILE="c:\sas\data\inventory.xls"
        DBMS=excel2000
        REPLACE;
RUN;
```

The DBMS option is used to specify what kind (and version) of the output file is to be created. If you have problems exporting a SAS data set into an excel file, note that DBMS=excel97 and DBMS=excel2002 specifications are also available; this example is set up for use with excel2000.

If the Excel file already exists, or if you aren't sure, include the REPLACE option; otherwise, SAS will not over-write it and thus not save the data set you want to export.

Note a Limitation on the Length of the File Name with PROC EXPORT. When PROC EXPORT places SAS datasets into Excel files, the length of the whole name of the Excel file, including the drive specification and path, cannot exceed 64 characters. Excel file names are usually "short", but a long path for the location of the file may be a reality. You should either change the location of the file or use the driver mapping to make the path shorter.

Exporting Datasets with Missing Data

Suppose you need to export a dataset with numeric and character columns containing missing values. However, the flat file should not contain the default dot or space used for missing values. Use either an empty string or the pattern NULL.

The Export Wizard defaults to the dot or space for the missing values. A way to override the dot or space when writing data to an comma separated value (CSV) file is to simply include missing=' ' in an OPTIONS statement. The character you type between the single quotes will replace the default missing character. For example:

```
OPTIONS missing='m';  /* Specify an m to be used for all Missing Values */

DATA temp;
```

```
SET class;
IF (weight > 250) THEN weight = . ;  /* Convert outlier to missing */
run;

PROC EXPORT DATA= work.temp
        OUTFILE= "c:\temp\test.csv"
        DBMS=CSV REPLACE ;
RUN;
```

Generate Excel Files with the Output Delivery System

Generating Excel spreadsheets containing SAS datasets using the Output Delivery System (ODS) is easy. ODS CSV writes a text file of values separated by commas and named with a CSV extension; ODS HTML writes a file with an XLS extension. Both files can be read by Excel as spreadsheets.

```
ODS CSV FILE=”c:\sas\data\class.csv”;

PROC PRINT DATA=class; RUN;
ODS CSV close;

ODS HTML FILE=”c:\sas\data\class.xls”;
PROC PRINT DATA=class NOobs; RUN;
ODS HTML close;
```

You can print variable labels to the Excel file if you add the Label option on the PROC PRINT statement. However, with this data transfer approach you cannot get both variable names and labels. The NOobs option omits the sequentially numbered column named ‘obs’ that SAS prints by default. A VAR statement will print selected variables in the order you specify rather than all variables in the dataset.

ODS does not actually produce Excel files since they are not an intended ODS destination; however, the files can be opened with Excel since it sees the registered extensions of .XLS or .CSV and recognizes these as valid file names to open.

GRAPHIC

SAS/GRAPH uses a series of PROCs, as do most SAS operations. Two of the basic PROCs are high-quality versions of 'regular' SAS PROCs that have always produced ASCII character-based plots and charts. The 'regular' PROCs are PROC PLOT and PROC CHART, and their SAS/GRAPH cousins are PROC GPLOT and PROC GCHART. Of course, then there are several SAS/GRAPH PROCs that do things that are not feasible in ASCII output, such as PROC GMAP, PROC CONTOUR and PROC G3D.

Let's consider a two-axis plot as our example of graphing real-world data. First, of course, we need a SAS Data Set containing the two variables we're going to graph. Say we want a plot of height vs. weight, to see if there is a trend toward taller people being heavier. Here is the DATA step (with the data entered instream):

```
data plotest;
  input height weight;
cards;
46  98
47  96
49 101
52 102
52 110
54 121
55 127
56 148
58 137
59 154
60 155
62 157
;
```

Now, if we want to plot these data in standard SAS output, we would run the following:

```
proc plot data=plotest;
  plot height*weight;
run;
```

We don't have many enhancement opportunities here, but we will see a plot in the output window.

To generate a similar plot in SAS/GRAPH, simply add one letter

```
proc gplot data=plotest;
  plot height*weight;
run;
```

The resulting graph is high-quality output, but will use all the defaults and is relatively boring. To spice it up, specify some colors and interpolation options and add a title. Notice the difference made by the enhancements added to the following program

```
symbol1 color=green v=star i=join;
title1 color=red 'Plot of Height versus Weight';
proc gplot data=plotest;
  plot height*weight;
run;
```

Imagine that your department keeps records of how many students come to the receptionist asking for information or help. After one year you have collected the following data:

```
data chartest;
  input month $ count;
cards;
SEP  87
OCT  94
NOV  98
DEC  67
JAN  42
FEB  79
MAR  92
APR  78
MAY  37
JUN  29
JUL  45
AUG  12
;
```

Now, if we want to chart these data in standard SAS output, we might run the following:

```
proc chart data = chartest;
  vbar month /  sumvar=count;
run;
```

The chart will be generated in the output window.There are a few enhancement options available. We could use MIDPOINT= to arrange the months in chronological, not alphabetical order, or SYMBOL= to ask SAS to use a character other than the default (*) when 'painting' the bar graph. But as usual, the enhancement options are slim. The following example and output show the previous program with the months re-ordered (chronologically) and the symbol changed from an asterisk (*) to a vertical bar (|).

```
proc chart data = chartest;
  vbar month / midpoints='SEP' 'OCT' 'NOV' 'DEC'
    'JAN' 'FEB' 'MAR' 'APR' 'MAY'
    'JUN' 'JUL' 'AUG' sumvar=count symbol='|';
run;
```

To generate a similar (but much more interesting) plot in SAS/GRAPH, simply remove the 'symbol=' keyword, add the letter 'g' to make it PROC GCHART.

```
title 'Bar Chart of Student Traffic';
proc gchart data = chartest;
  vbar month / midpoints='SEP' 'OCT' 'NOV' 'DEC'
    'JAN' 'FEB' 'MAR' 'APR' 'MAY'
```

```
    'JUN' 'JUL' 'AUG'  sumvar = count;
 run;
```

This graph will be high-quality output, but will still be relatively straightforward and simple. Here's where the flexibility of SAS/GRAPH is very handy. We can change the look of the 'fill' of each bar, add colors, put a frame around the bar portion of the graph, and add the actual number at the top of each bar.

To change the fill pattern and add colors to the bars, use PATTERN statements. To put a frame around the bars, add the FRAME option after the slash in the VBAR statement. To place the actual number at the top of each bar, specify the SUM option, also in the VBAR statement after the slash. PATTERNID=MIDPOINT requests that SAS change the patterns, according to the PATTERNn statements, for each successive bar in the graph. In our example this has the effect of highlighting each semester (i.e., each group of four months).

```
 pattern1 value = X3 color = magenta repeat=4;
 pattern2 value = R5 color = green repeat=4;
 pattern3 value = L2 color = blue repeat=4;
 title1 'Bar Chart of Student Traffic';
 proc gchart data = chartest;
  vbar month / midpoints='SEP' 'OCT' 'NOV' 'DEC'
    'JAN' 'FEB' 'MAR' 'APR' 'MAY'
    'JUN' 'JUL' 'AUG'  sumvar = count
    frame sum patternid = midpoint;
 run;
```

Now let's look at a graphic that is almost useless in standard output, but quite presentable in SAS/GRAPH. We'll use PROC CHART again, this time creating a primitive star chart (a variant of a pie chart). We use the PAGESIZE option to give SAS sufficient room (45 lines) to chart the star.

```
 options pagesize=45;
 proc chart data = chartest;
  star month / midpoints='SEP' 'OCT' 'NOV' 'DEC'
    'JAN' 'FEB' 'MAR' 'APR' 'MAY'
    'JUN' 'JUL' 'AUG'  sumvar = count;
 run;
```

Now let's enter the real world, running the same program with PROC GCHART. This time we use no patterns, so we can see the cells more easily. Colors can be added later.

```
 title1 'Star Chart of Student Traffic';
 proc gchart data = chartest;
  star month / midpoints='SEP' 'OCT' 'NOV' 'DEC'
    'JAN' 'FEB' 'MAR' 'APR' 'MAY'
    'JUN' 'JUL' 'AUG'  sumvar = count;
```

```
  run;
```

http://support.sas.com/techsup/sample/sample_graph.html

http://www.sas.com/technologies/bi/query_reporting/graph/