

Graphs Without SAS/GRAPH^{AE}

Howard Schreier, U.S. Department of Commerce, Washington, DC

ABSTRACT

The task the paper tackles is to generate a large number of simple charts for a web site under the situation where SAS/GRAPH software is not available. The solution employed is to use Microsoft Excel instead, and have SAS^{AE} Base software drive the process via Dynamic Data Exchange.

INTRODUCTION

The technique described and illustrated in this paper is one solution to the problem. It assumes that SAS Base Software is the tool of choice for the underlying data management. The environment is Windows 98, SAS 8.1 and Excel 2000 (but the method does not depend on especially new features in any of these products).

The technique uses SAS to organize and manage the process, Excel to produce the charts and Dynamic Data Exchange (DDE) to communicate between SAS (as the client) and Excel (as the server).

The solution is embodied in two components, an Excel workbook template and a SAS DATA step, carefully designed to work together.

THE EXCEL TEMPLATE

The template includes several model charts; each of these locks in nearly all of the design characteristics (for example, background color), at least from the perspective of the SAS client. However, a few "hooks" are left to permit data-driven modifications at runtime. It also contains a VBA (Visual Basic for Applications) macro which makes a copy of one model chart and connects it with a set of data points and other specifications, then exports the result as a GIF (Graphics Interchange Format) file. Finally, the template provides work space (comprising named worksheets and named data ranges); these names are used in both the SAS code and the VBA code, providing common ground for the purpose of coordination.

THE SAS DATA STEP

The SAS code processes data sets containing the plot points and other specifications to be applied to the charts at runtime. It generates a series of commands which are sent to Excel via the DDE link. BY group processing is used to implement production of multiple charts.

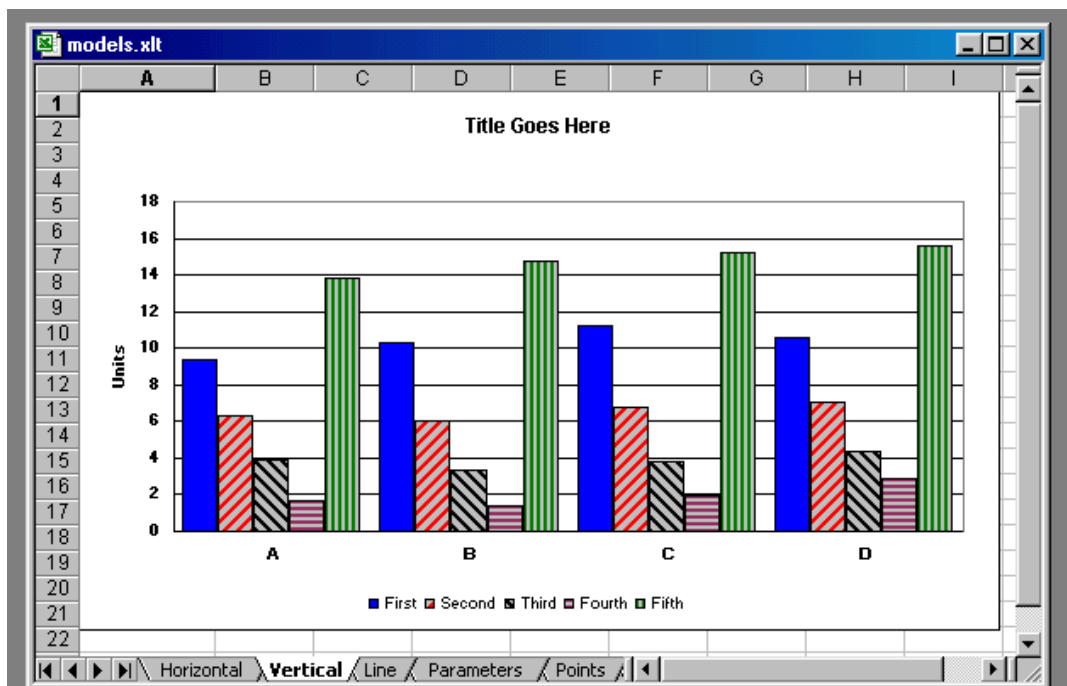
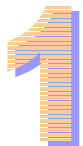
MORE ABOUT THE EXCEL TEMPLATE

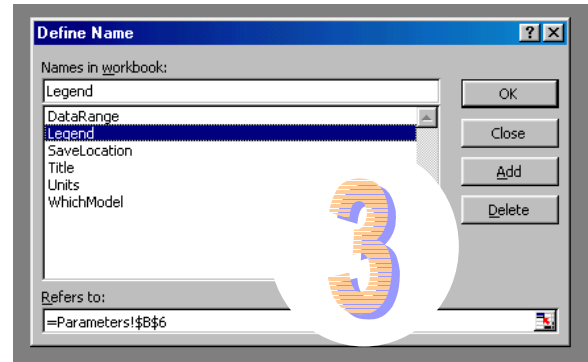
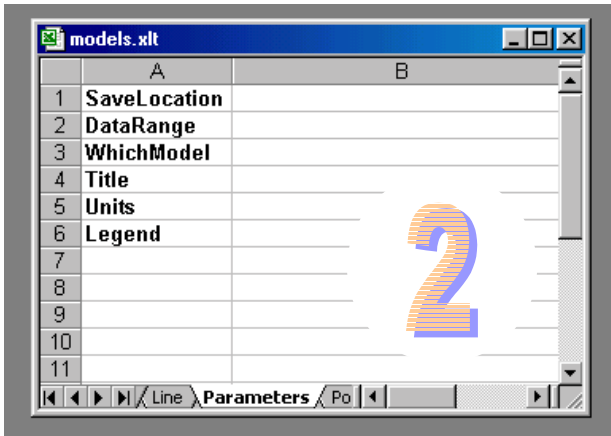
The workbook template (models.xls) contains six worksheets and one VBA module. Three of the worksheets each contain one model chart:

- "Vertical": a column chart for up to five series. See Figure 1.
- "Horizontal": a bar chart for one series
- "Line": a line chart for up to five series

The fourth worksheet (named "Parameters") contains a series of single-cell named ranges in column B. These are used to temporarily store information, recorded in response to commands from SAS, where it can be retrieved by the VBA routine. For example, a "Y" or "N", indicating whether or not a legend should appear, is stored in cell B6, which has been given the name "Legend". Purely for documentation, the name of each range is stored in the adjacent cell in column A. See Figures 2 and 3. Two other worksheets are initially empty and are used and then cleared within each BY-group cycle.

- "Points" is where the data grid (values to be plotted and the associated labels) is stored.
- "Live Chart" is where a copy of one of the model charts is pasted, then modified and exported.





4

```
chart_id=102 type=Line series=3 legend=Y
title=U.S. exports: Automotive tires and tubes
units=Millions of dollars
sname1=Canada sname2=Mexico sname3=Other sname4= sname5=
dirname=nafta filename=30220
```

```
chart_id=185 type=Horizontal series=1 legend=N
title=U.S. 2000 exports: Chemicals-fertilizers
units=Millions of dollars
sname1= sname2= sname3= sname4= sname5=
dirname=top10 filename=12510
```

5

Obs	chart_id	category	s1	s2	s3	s4	s5
1213	102	1989	232	59	395	.	.
1214	102	1990	385	88	423	.	.
1215	102	1991	439	135	470	.	.
1216	102	1992	470	158	524	.	.
1217	102	1993	512	176	505	.	.
1218	102	1994	551	169	580	.	.
1219	102	1995	652	176	733	.	.
1220	102	1996	638	236	771	.	.
1221	102	1997	735	444	879	.	.
1222	102	1998	797	549	791	.	.
1223	102	1999	946	461	632	.	.
1224	102	2000	1016	475	579	.	.
2112	185	Canada	744
2113	185	China	683
2114	185	Brazil	379
2115	185	Japan	257
2116	185	Australia	248
2117	185	Mexico	247
2118	185	Korea	173
2119	185	Argentina	116
2120	185	India	103
2121	185	Belgium	88

6

```
data _null_;
file commands;
merge designxp.design pointsxp.points; by chart_id;
if first.chart_id then do;
  * Clean out any data from previous chart;
  put '[WORKBOOK.SELECT("Points")]';
  put '[SELECT("C1:C6")]';
  put '[CLEAR(3)]';
  * Load series labels (for chart legend);
  put '[WORKBOOK.SELECT("Points")]';
  put '[SELECT("R1C1")]';
  array sname(5) sname1-sname5;
  do snum = 1 to series;
    put '[SELECT("RC[1]")]';
    put '[FORMULA("'" sname(snum) +(-1) "')]';
  end;
  rownum = 1; * so data will load in rows 2, 3, ...;
end;
* Load data from each observation into worksheet;
rownum + 1;
* Label, for category axis;
put '[SELECT("R' rownum +(-1) 'C1')]';
put '[FORMULA("'" category +(-1) "')]';
* Numeric;
array s(5) s1-s5;
do snum = 1 to series;
  put '[SELECT("RC[1]")]';
  put '[FORMULA("'" s(snum) +(-1) "')]';
end;
if last.chart_id then do;
  * The data grid has been completed. The technique for passing other
  parameters is to write them into named single-cell ranges which have
  been pre-defined in the "Parameters" worksheet. First, remove any
  residue.;
  put '[WORKBOOK.SELECT("Parameters")]';
  put '[SELECT("C2")]';
  put '[CLEAR(3)]';
  * Extent of data, in "A1" style (eg, A1:C11);
  colnum = series + 1; * allows for category labels;
  colalpha = substr('ABCDEF',colnum,1);
  put '[WORKBOOK.SELECT("Parameters")]';
  put '[SELECT("DataRange")]';
  put '[FORMULA("A1:' colalpha +(-1) rownum +(-1) "')]';
  * Choice of which model chart to use. Value must correspond to
  a worksheet which exists in the template;
  put '[SELECT("WhichModel")]';
  put '[FORMULA("'" type +(-1) "')]';
  * Title;
  put '[SELECT("Title")]';
  put '[FORMULA("'" title +(-1) "')]';
  * Description of units;
  put '[SELECT("Units")]';
  put '[FORMULA("'" units +(-1) "')]';
  * Legend toggle (Y/N);
  put '[SELECT("Legend")]';
  put '[FORMULA("'" legend +(-1) "')]';
  * Location for GIF containing the finished chart;
  xlfile = "&OUTPATH." || trim(dirname) || '\' || filename;
  put '[SELECT("SaveLocation")]'; put '[FORMULA("'" xlfile +(-1) "')]';
  * All information has been transferred, so run the VBA subroutine
  which actually produces the chart;
  put '[RUN("GenChart")]';
end;
run;
```

MORE ABOUT THE SAS DATA STEP

Before examining the code itself (genchart.sas) it's helpful to inventory what must be present in the environment for it to run successfully. The list can be subdivided into three pieces: Excel, SAS data, and the host system filespace.

Excel must be running, with the template (or a workbook constructed from the template) open and active. The fileref "COMMANDS" must be assigned to the "System" topic of Excel's DDE server; the code for this is

```
filename commands dde 'excel|system';
```

Two SAS data sets, containing the information needed to make the individual graphs, must be accessible.

- DESIGN contains variables which pertain to an entire chart: which of the model charts to use, the number of data series, whether or not a legend is to be generated, a title, a label for the value axis, labels for each of the series, the directory name and file name for the output. See Figure 4 for a display of two representative observations. There must be a libref "DESIGNXP" pointing to a library containing this data set.
- POINTS contains variables which pertain to one level along the category axis, namely a label for that level and a value for each series. See Figure 5 for a display of two representative BY groups (corresponding to the data in Figure 4). There must be a libref "POINTSXP" pointing to a library containing this data set.

Both data sets contains a common variable (CHART_ID), used to merge.

There must be a macrovariable "OUTPATH" designating an absolute path in the host filespace suitable for storage of the output files. There must be a subdirectory relative to that path for each value appearing in the DIRNAME variable.

The author has prepared a self-contained demonstration package providing all of the ingredients (with the obvious exception of the Excel and SAS software).

The DATA step (see Figure 6) is fed by a MERGE statement which combines the two data sets. The BY variable (CHART_ID) not only links the two data sets, but also defines BY-group boundaries in the processing. There is a one-to-one correspondence between these BY groups and the charts to be produced.


The DATA step follows a very commonly encountered structure. There is a block of code conditional on FIRST.CHART_ID which does some initial setup for each chart; in particular, it cleans up the space in the Excel workbook where the data for the chart will be written. The block of code which is unconditional then fills in this space with data. Finally, the block of code conditional on LAST.CHART_ID passes other parameters to Excel and then tells Excel to run the VBA code which actually generates the chart.

The output of the DATA step is not a file, but rather a stream of character strings sent to the DDE "System" topic through which Excel accepts commands. The commands must be valid in terms of the Excel Version 4 macro language.

THE COMMAND STREAM

Figure 7 presents the portion of the stream of commands which corresponds to one of the two data sequences (CHART_ID values) in Figures 4 and 5. Part of a series of highly repetitious lines is omitted to save space, but one can still read through and discern the sequence of events:

- Worksheet "Points" is cleared of data.
- Series labels are recorded along the top row of "Points".



```
[WORKBOOK.SELECT("Points")]
[SELECT("C1:C6")]
[CLEAR(3)]
[WORKBOOK.SELECT("Points")]
[SELECT("R1C1")]
[SELECT("RC[1]")]
[FORMULA("Canada")]
[SELECT("RC[1]")]
[FORMULA("Mexico")]
[SELECT("RC[1]")]
[FORMULA("Other")]
[SELECT("R2C1")]
[FORMULA("1989")]
[SELECT("RC[1]")]
[FORMULA("231.582759")]
[SELECT("RC[1]")]
[FORMULA("59.244973")]
[SELECT("RC[1]")]
[FORMULA("395.457595")]
[SELECT("R3C1")]
[FORMULA("1990")]
[SELECT("RC[1]")]
[FORMULA("384.528677")]
[SELECT("RC[1]")]
[FORMULA("87.706213")]
[SELECT("RC[1]")]
[FORMULA("422.709764")]

. . .

[SELECT("R13C1")]
[FORMULA("2000")]
[SELECT("RC[1]")]
[FORMULA("1015.85672")]
[SELECT("RC[1]")]
[FORMULA("474.978904")]
[SELECT("RC[1]")]
[FORMULA("579.291535")]
[WORKBOOK.SELECT("Parameters")]
[SELECT("C2")]
[CLEAR(3)]
[WORKBOOK.SELECT("Parameters")]
[SELECT("DataRange")]
[FORMULA("A1:D13")]
[SELECT("WhichModel")]
[FORMULA("Line")]
[SELECT("Title")]
[FORMULA("U.S. exports: Auto . . .")]
[SELECT("Units")]
[FORMULA("Millions of dollars")]
[SELECT("Legend")]
[FORMULA("Y")]
[SELECT("SaveLocation")]
[FORMULA("c:\stage\nesugcharts\ NAFTA\30220")]
[RUN("GenChart")]
```

- Subsequent rows of "Points" are filled with data, category labels in Column A ("C1" in Excel's alternative notation) and numeric values in other columns.
- Column B (or "C2") of Worksheet "Parameters" is cleared of data
- Individual cells are selected (by name, not by coordinates) and filled with various pieces of information. For example, the legend toggle ("Y" or "N") is stored in B6 ("Legend").
- The VBA routine ("GenChart") is launched.

THE VBA CODE

The VBA routine (GenChart; see Figure 8)

- cycles through the names for the parameters; makes the appropriate one-cell range active and transfers its value to a local variable with the same name
- makes a copy of the appropriate model chart
- modifies the data reference in this newly created chart, so that it points to the precisely correct data grid
- toggles the legend on or off and applies the chart title and the value axis label (units of measure)
- exports the chart to the designated location
- clears the temporary chart in preparation for the next cycle

Each time it runs, GenChart creates a GIF file containing a chart. Figures 9 and 10 are web-browser renderings of the charts produced from the data in Figure 5.

ADDITIONAL COMMENTS

Why use Excel? Primarily, it is a good choice because it has the necessary functionality. A second, and very important consideration, is that there is a substantial accumulated literature (in SAS user group conference proceedings and in the SAS-L archives) on the interoperation of SAS and Excel. That cannot be said about any other competing product.

The two Excel macro languages. The method illustrated here uses both VBA and the Excel V.4 macro language (which is how the commands sent from SAS to Excel via DDE are expressed). Both are involved out of necessity, since Excel's DDE server does not understand VBA while only VBA can operate all of the features of recent versions of Excel. Another consideration is that the macro recorders in recent versions of Excel generate only VBA. Using the recorder can be a tremendous help, especially for the novice coder.

DDE sophistication. Note that the DDE used here is very simple. It's one-way (nothing is passed from Excel to SAS, only from SAS to Excel) and relies on the System topic alone. The more common way to pass data via DDE is to set up a data buffer which can transfer a large rectangular range of cells at once. Instead, SELECT and FORMULA commands are used here to fill one cell at a time. It is much slower that way, but the volumes of data are so tiny that the difference is negligible, and worth the simplicity gained.

Performance and scalability. The author has used this technique to produce several thousand charts in one

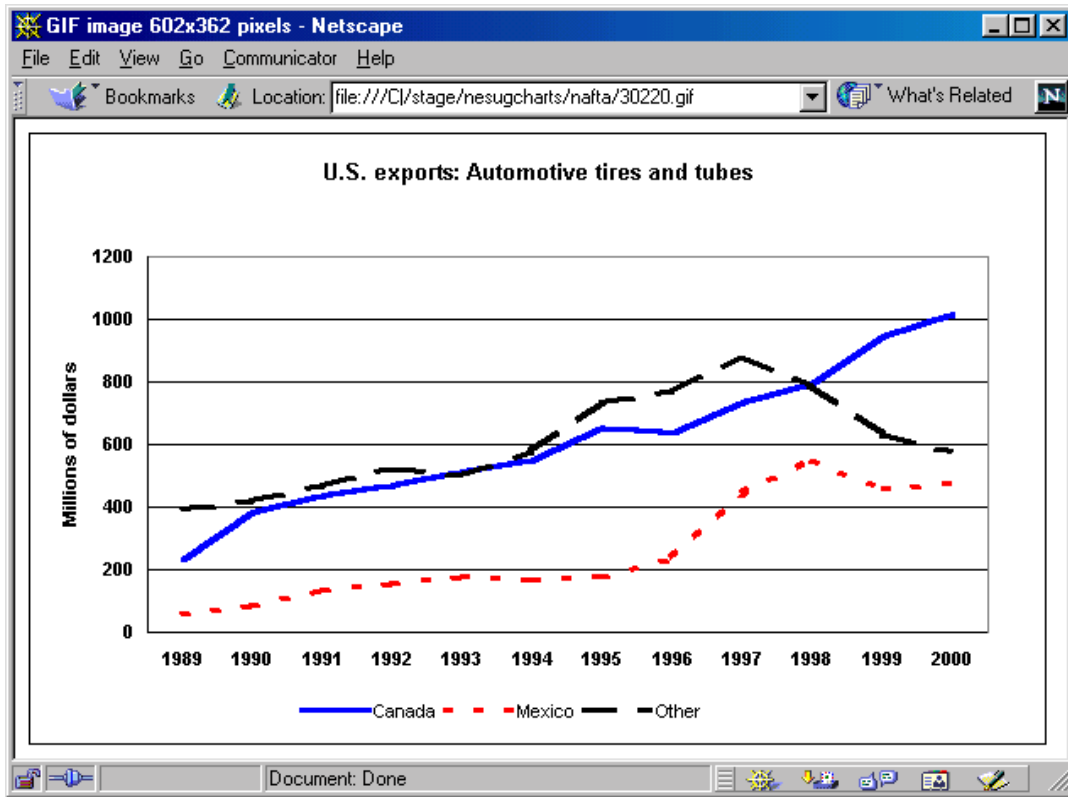
```

Sub GenChart()
'Pick up range of data points (including labels)
  Dim DataRange As String
  Application.Goto Reference:="DataRange"
  DataRange = ActiveCell.Value
'Pick up choice of chart type
  Dim WhichModel As String
  Application.Goto Reference:="WhichModel"
  WhichModel = ActiveCell.Value
'Pick up title
  Dim Title As String
  Application.Goto Reference:="Title"
  Title = ActiveCell.Value
'Pick up legend toggle value
  Dim Legend As String
  Application.Goto Reference:="Legend"
  Legend = ActiveCell.Value
'Pick up scale (units) description
  Dim Units As String
  Application.Goto Reference:="Units"
  Units = ActiveCell.Value
'Pick up target path and filename
  Dim SaveLocation As String
  Application.Goto Reference:="SaveLocation"
  SaveLocation = ActiveCell.Value
'Make a copy of the model chart
  Sheets(WhichModel).Select
  ActiveSheet.ChartObjects(1).Activate
  ActiveChart.ChartArea.Select
  ActiveChart.ChartArea.Copy
  Sheets("Live Chart").Select
  ActiveSheet.Paste
'Reset data references
  ActiveChart.ChartArea.Select
  ActiveChart.SetSourceData _
    Source:=Sheets("Points").Range(DataRange), _
    PlotBy:=xlColumns
'Take care of title, axis label, legend
  With ActiveChart
    .HasTitle = True
    If Legend = "Y" Then
      .HasLegend = True
    Else
      .HasLegend = False
    End If
    .ChartTitle.Characters.Text = Title
    With .Axes(xlValue, xlPrimary)
      .HasTitle = True
      .AxisTitle.Characters.Text = Units
    End With
  End With
'Export updated chart as a GIF
  ActiveChart.Export _
    Filename:=SaveLocation & ".gif", _
    FilterName:="gif"
'Clean up
  ActiveChart.ChartArea.Clear
End Sub

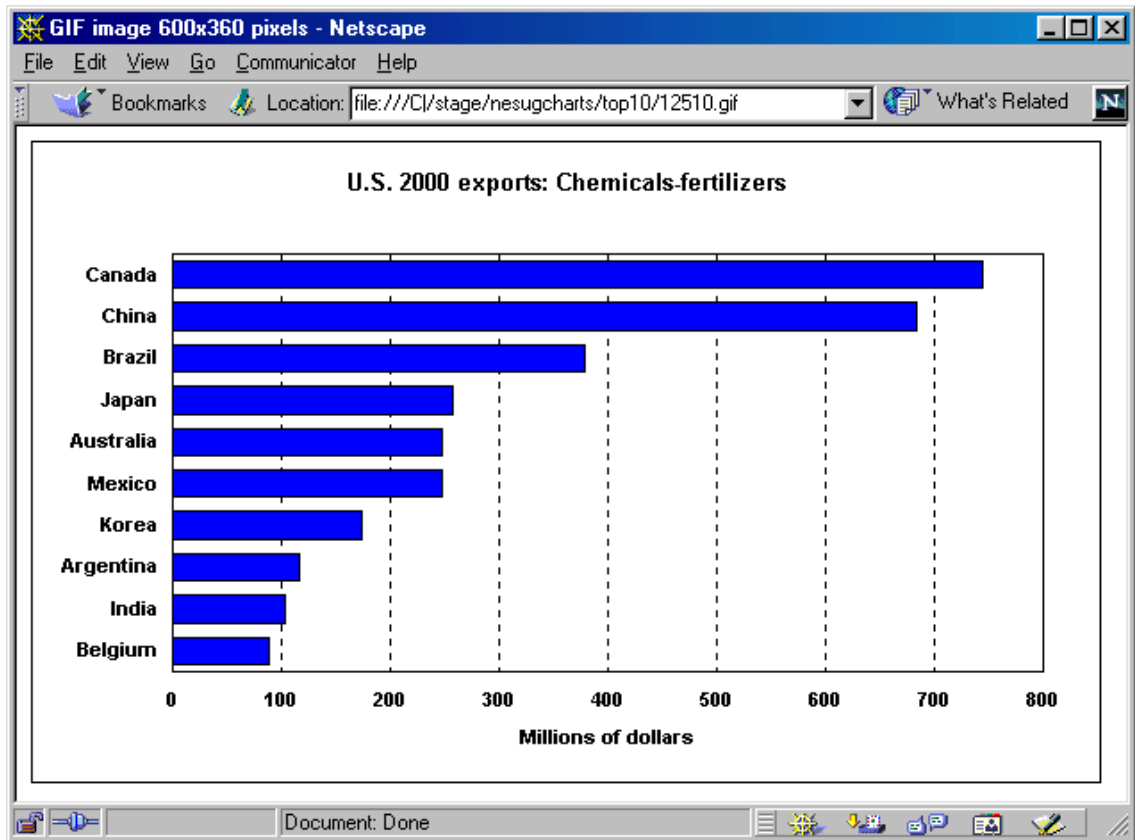
```



9



10



run. On a Pentium III desktop system, it took about one second per chart. While parts of the code could be streamlined, there remains the overhead and latency introduced by the need to interact with the host's file system and create a new file for each chart. In theory, if one does not mind letting a computer grind away for hours, it should be possible to scale even higher. But using a host's file system to manage charts in such numbers may not be desirable. Alternatives, for web content, include generating charts on-the-fly as they are requested or storing them as binary objects in a data base.

NEXT STEPS?

The working code shown here is basically just a proof of concept and a test of scalability. What enhancements might be added to create a good production system?

The present code allows the SAS program to control only a few characteristics of the chart. For example, the legend can be toggled on or off. It is possible to expose many more Excel chart options and attributes at runtime. The list of specifications stored in the "Parameters" worksheet would then grow, and there would be added SAS code to record them and added VBA code to implement them. For example, one could toggle the grid lines or even change things like their color and weight. This would make the facility much more flexible and general. In principle, we don't need really need the Excel template at all. Instead, we could start with an empty workbook, create charts from the built-in types and pass all design details from SAS.

A SAS macro could encapsulate the process of generating the DDE command stream.

There could be global design choices, which would apply to (or at least be in effect by default for) all of the charts produced in a run. For example, the colors of the axes might be specified at this level. A convenient way to enter these would be via a front-end on-screen form, which could be also be used for selecting data sources and output destinations.

The facility could be much more robust. The code shown here does virtually no error checking. For example, if the SAS variable TYPE takes on a value which does not correspond to the name of a worksheet containing a model chart (such as "Vertical"), it causes a rather ugly runtime error.

CONCLUSIONS

Is this an adequate substitute for SAS/GRAPH? Not really, if your data live in SAS, or if SAS is your tool of choice for data work. SAS can talk to Excel via DDE, but it's far from seamless.

To produce a small number of one-off charts, this technique is

probably more trouble than it is worth. Instead, one can simply use manual methods to export the data from SAS, then import in Excel and develop the charts strictly within the Excel user interface. The development of this DDE-based technique was motivated by the need for an automated process for mass production of a large number of essentially similar charts, and it promises to be a reasonably adequate substitute for SAS/GRAPH in that situation.

REFERENCES

Microsoft Corp. (1992), *Function Reference / Microsoft Excel, Version 4.0*, Redmond, WA: Microsoft Corp.

Roman, Steven (1999), *Writing Excel Macros*, Sebastopol, CA: O'Reilly & Associates, Inc.

SAS Institute Inc. (1999), *SAS Companion for the Microsoft Windows Environment, Version 8*, Cary, NC: SAS Institute Inc.

Vyverman, Koen (2001), "Using Dynamic Data Exchange to Export Your SAS^Æ Data to MS Excel; Against All ODS, Part I", *Proceedings of the Twenty-sixth Annual SAS Users Group International Conference* (<http://www2.sas.com/proceedings/sugi26/p011-26.pdf>)

SAS-L (<http://www.listserv.uga.edu/archives/sas-l.html>)

Microsoft Excel Charting Newsgroup
(<news://msnews.microsoft.com/microsoft.public.excel.charting>)

CONTACT INFORMATION

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

Howard Schreier
U.S. Department of Commerce
H-2815
Washington DC 20230
(202) 482-4180
Fax: (202) 482-4614
Howard_Schreier@ita.doc.gov

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. ^Æindicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.