

Paper 15-25

**SAS[®] SOFTWARE AND MICROSOFT OFFICE VISUAL BASIC FOR APPLICATIONS
MAKE BEAUTIFUL REPORTS TOGETHER****Aileen L. Yam, PharmaNet, Inc., Princeton, NJ****ABSTRACT**

In pharmaceutical research, clinical study reports contain word-processing tables that are usually retyped from SAS output. In order to rebundle the data into a tabular format or into a more refined look and to ensure accuracy, a considerable amount of manual labor in typing and verifying is involved. Whenever data values change, the process is to be repeated. On the other hand, if dedicated programs or report templates are created to output every word-processing table programmatically, there will be a substantial amount of programming development efforts as well as a long list of report designs to store. This paper describes the main points in developing a general-purpose Visual Basic for Applications program for repackaging a wide variety of output. The key is to create user-defined controls in the SAS programs.

Introduction

If a staff of secretaries and proofreaders is not available, or if there is no word-processing or publishing department, or if you are a medical writer or a statistician who enters numbers into tables and verifies the numbers in clinical study reports, or if you are a SAS programmer who wants to create tabular reports in a PC document-based environment, or if you are a Visual Basic for Applications (VBA) programmer looking for an automated way to take care of these needs, you may be interested in reading this paper.

Development Philosophy

SAS output in the form of .lst files are space-delimited and not tab-delimited files. If fixed-spaced fonts such as Courier New or SAS Monospace are used, the columns will line up well when the SAS .lst files are converted to Microsoft Word documents. To change fonts, to align data, to add borders around the .lst files, and to add lines between the columns or across the rows of the .lst files are not simple jobs, because the .lst files are not spreadsheets where rows and columns can be referenced directly. Since

powerful VBA macros in Microsoft Excel can be developed, and there are procedures such as PROC ACCESS or DDE that can transfer data from SAS to Excel, and Excel can interface with Word via Object Linking and Embedding (OLE), Excel serves as an intermediate for repackaging SAS data into Word documents.

Reformatting the .lst files in Word without going through Excel seems to be a more direct approach, but it involves the use of bookmarks to do the reformatting. Bookmarks rely on specific location and data values. If the specific information changes, the reformatting is no longer correct. In addition, the process needs a customized macro for each customized table.

Although it is easier to write a customized macro for each kind of table, there will be either a narrowly defined set of choices or a long list of macros associated with different report designs to keep track of. In addition, it will take a team of VBA programmers to develop and maintain many customized macros. Thus, the goal is to build a fairly comprehensive Excel VBA macro that is applicable to various table formats and reusable for many projects.

When a specific VBA program is written to output a specific word-processing table, the programming code is specific. It has specific commands on when and where to put borders, to draw lines, to select certain fonts, or to align specific columns. By turning the specific VBA commands into user-defined variables from SAS data, the SAS users, and not the VBA macro, are in control. The SAS users will then have the flexibility to design the tables to their liking without investing time in learning Visual Basic.

Development Methodology

Each SAS program is set up with the same underlying method: 24 SAS variables are created for specifying formatting instructions. The names and the location of the 24 variables are chosen to make the data set-up and report building in SAS as intuitive as possible.

The first SAS variable created is SEQ. For a small table that fits within one page, SEQ is the table type and table number, such as Table 1 or Appendix 1. For a large table that takes up more than one page, SEQ consists of table type, table number and page number. For example, table 3 has 2 pages, so the variable SEQ for table 3 will have the values of Table 3.1 and Table 3.2.

The second variable, OBSNUM, contains the observation numbers of each row of data, derived from the SAS automatic variable `_n_`, plus any number of titles, column labels, or footnotes inserted later.

The third variable, LINE, has the value of 1 or 2 to indicate whether a single line or a double line is to be drawn across the bottom of an observation.

The fourth variable, HEADFOOT, indicates that the observation is a title if HEADFOOT has a value of 1, or a column label if HEADFOOT has a value of 2, or a footnote without page number if HEADFOOT has a value of 8, or a footnote with page number if HEADFOOT has a value of 9.

The fifth variable, FONT, is for specifying bold font with a value of B, or italic font with a value of I, or bold italic font with BI. To add spaces to a particular row of text in the first column, assign the value S followed by the number of spaces to the variable FONT. For example, to add 8 spaces, use

S8. To make the text bold italic and to add 8 spaces, use BIS8.

The sixth variable, FONTYPE, is for specifying the type and the size of font, for example, Arial 8. The font size of table numbers and titles are automatically made a little larger than the font size specified for the rest of the document. The type of font selected needs to be available in the installation of Word and needs to be spelled exactly the same as in the font list. Although the font style can vary from row to row as specified in the fifth variable, the font type and size remain constant for the whole table.

The seventh variable, ALGCHG is for changing alignment in numeric or character values. Excel has a tendency to interpret data. For example, if the age range 12-65 is entered into a cell, Excel automatically interprets it as a date value 12/1/1965 and formats the value as Dec-65. To prevent Excel from interpreting numeric data values, all the cells are formatted as text before data are entered. The default alignment for all the values is set to be centering. ALGCHG is a character variable with the values 'L' for left justifying, 'R' for right justifying, 'C' for centering, 'T' for aligning vertically to top, and 'B' for aligning vertically to bottom. The vertical alignments are applicable to merged cells only. The letter 'D' is for centering numbers and yet aligning the numbers by decimal, or if there is no decimal, centering and aligning integers by the last digit. The letter 'P' is for aligning numbers that contain parentheses and percent signs such as 123 (45.6%).

The beginning rows and columns and the ending rows and columns that are to be aligned are defined in the next four variables, ALIGNBR, ALIGNBC, ALIGNER and ALIGNEC. It is clearer, but not necessary, to specify the beginning and the ending rows and columns on the same observation as where the cells are to be aligned. The Excel VBA macro scans every observation of the SAS data to take action.

The variables for adding a line to the cells that are being merged, and the beginning and ending rows and columns that are to be merged as a single cell can be specified in any of the next two sets of five variables, MADDLIN1, MERGEBR1, MERGEBEC1, MERGEER1, MERGEEC1, and MADDLIN2, MERGEBR2, MERGEBEC2, MERGEER2, MERGEEC2. These two sets are created because one set is not enough to handle all the merging tasks. In the Excel VBA macro, the default horizontal

alignment after merging is centering across the merged cells, the default vertical alignment after merging is putting the text of the merged cells at the bottom. Merging cells causes the line below the merged cells to disappear if there is a line. The MADDLINE1 and MADDLIN2 variables tell the Excel VBA macro whether a line exists and needs to be added back after the merging occurs. The data values that appear in the table after merging need to be put at the upper-left most cell of all the cells that are to be merged, otherwise the data values will disappear after merging. If the cells that are to be merged contain data values in more than one cell, a warning message appears:

The selection contains multiple data values. Merging into one cell will keep the upper-left most data only.

This is a characteristic of the merge cells routine in Excel. Putting data values in the upper-left most cell of all the cells that are to be merged is made a requirement in setting up the SAS data, and the warning message is suppressed in the Excel VBA macro.

The twenty-second variable, P_L is for indicating whether the table will be portrait or landscape.

The twenty-third variable, HFTEXT, is for storing table numbers, titles and footnotes, which are usually long and have to be formatted differently, so this extra variable is needed.

The twenty-fourth variable, TOTVAR, is for indicating the total number of columns in a table.

If the third to the twenty-first variables have the values of zero for numeric variables or null for character variables, then no formatting needs to be done. All these variables for formatting tables are initially set as zero for numeric variables or blank for character variables with the RETAIN statement. By default, no formatting is done, unless specified later in the SAS programs by the users.

Other variables with the prefix VAR are created to standardize the input data for outputting into tables. The total number of VAR variables created is based on the total number of columns needed in a table.

PROC SQL with the INSERT statement is used to add extra observations to a table, such as titles and column labels.

For tables with more than one page, the CALL EXECUTE statement is used to output table numbers with the concatenated text string (cont'd), and to add a page extension to the variable SEQ for sorting purposes. The first page has the extension of .1, the second page has the extension of .2, and so on.

Conditional footnotes can be made by creating a Boolean variable to indicate when the footnote is needed.

Paging is controlled by the following statement:

```
page = ceil(_n_ / &pagesize);
```

where &pagesize is a macro variable for specifying the total number of rows on a page.

A macro with an ATTRIB statement is included in each SAS program to standardize the format and the order of the 24 variables. The format and order of the variables are made uniform so that data will not be truncated or cause other errors when concatenated.

Both the names and the order of the 24 variables are fixed. They are vehicles for communicating to the Excel VBA macro what needs to be done.

The data for many tables can be concatenated into a single SAS data set. The Excel VBA macro sorts the data by table number and page number and outputs each table or each page of a multi-page table into a separate page in Word.

Visual Basic for Applications Macro in Excel

The VBA program residing in Excel as Excel macro takes instructions from the 24 SAS variables to create tables in Excel and the tables are transferred to Word documents with OLE. The magic number 24 is chosen to strike a balance between completeness and ease of use. The objective is to make the Excel VBA macro complete enough to handle most aspects of complex formatting, and yet would not cause it too cumbersome to use.

Some defaults are set-up in the Excel VBA macro:

1. all input data are centered within columns;
2. the horizontal alignment of merged cells is centering across all merged cells;

3. the vertical alignment of merged cells is putting the data from the top-left most cell to the bottom of all merged cells;
4. an outside border is drawn around the four sides of a table;
5. an inside vertical border is drawn between each data column;
6. all the input data are sorted by the values of the first two SAS variables, SEQ and OBSNUM, so that the output tables generated in the Word documents are automatically sorted;
7. all the column widths are automatically adjusted to achieve the best fit.

The first three defaults can be modified by supplying different values in the formatting variables in SAS programs. The last four items are made defaults based on common sense as well as the most popular table formats from client specifications. To control the amount of complexity, no SAS variables are created to change the last four defaults. Changes to the last four defaults can only occur by modifying the Excel VBA macro.

Before running the Excel VBA macro, all the Microsoft Word and Microsoft Outlook applications or files need to be closed, because OLE automation is implemented to copy the reports from Excel to Word. Both Word and Outlook use Word objects and there will be error message on automation error if Word or Outlook application or file is already open.

The Excel VBA macro has features for error and event handling, such as:

1. When the Excel VBA macro is open, a reminder message appears to ask users to close any open Word or Outlook application or file in order to avoid automation errors.
2. When a user fills in a dialog box to select and open a data file, there was logic written to detect if the data file has the correct structure. If a file with incorrect structure is open, warning appears, and users can select another file or cancel and quit;
3. Users enter table sort order into the dialog box, and the sort items can be cleared if typing mistakes are made;
4. If the sort item names cannot be found in the data file, or if the data files have more sort item types than those entered, warning is given and users can retype the names;
5. If the QUIT button is clicked, users are asked for confirmation;

6. The appearance of the pointer is changed from I-beam to hourglass when the Excel VBA macro is running. When the macro has finished running, the pointer is reset to I-beam, and a message 'Job is Done!' appears.

The overall font type and size can be changed in the FONTCHANGE module in the Excel VBA macro. Symbol fonts are marked before the font change occurs and the changed fonts are reverted back to symbol fonts at places where marked.

To maintain data integrity, data values and the number of decimal places are not changed in Excel or Word.

Sorting the tables is an elaborate procedure. The tables are sorted by the variables SEQ and OBSNUM, which contain table type, table number, and observation number. SEQ is parsed into smaller chunks, first separating table types from table numbers, and then further separating the table numbers into smaller units if there are extensions or alphabets in the table numbers. Table types are sorted in an order inputted into a dialog box by the user, e.g. Table before Appendix. Logic was written to compare user input with data to make sure all table types are accounted for, and to give warning if discrepancy between user input and data is found. Each small unit of table numbers is sorted to obtain numeric sorting where possible. If table numbers with extensions and alphabets are sorted as a single unit, then the sorting becomes alphabetical and may not always be correct, e.g., 10.1 will appear before 2. When breaking the table numbers into smaller units, the maximum number of units is counted and zeroes are temporarily padded into the table numbers that have less units, because Excel sorts blanks last, and zeroes will cause the blanks to be sorted first. By default, Excel software only allows 3 levels of sorting, but with table types, table numbers broken down into small units, and observation numbers, there are more than 3 units. To get around this limit, a DO Loop is used and the sorting is done one unit at a time in reverse order of importance, i.e. the least important unit gets sorted first.

In Word, there is an option for decimal tab, so numbers in a column can be centered and at the same time aligned by decimal. In Excel, the feature of decimal tab is not present, so the biggest challenge in writing the Excel VBA macro is to create an algorithm to serve the function of decimal tab when the numbers in a column need to be centered as well as properly aligned.

Another major challenge is to write logic in the Excel VBA macro to make intelligent guesses on the best fit in widths of each cell and each column of data. If there is no need to merge cells across columns, Excel's AUTOFIT feature can do the job. But the merge cells procedure can make the some of the column widths look unbalanced, or can cause some of the text to overflow. All possible merge cells scenarios need to be mapped out, and built into the programming logic. Just imagine you were to create 100 tables and had to fit the columns by hand through trial and error, so you decide to write a general program that automatically fits the columns under any circumstance precisely, and you can picture the amount of efforts involved.

Conclusion

The key to automate clinical study report production is — let SAS users and the SAS data drive the solution for creating a single Excel VBA macro to generate any Word table.

This is the second of a series of papers on the integration of SAS software with Visual Basic for Applications. The first paper was on using SAS software and VBA to mass-produce customized PowerPoint graphs. Please see the reference below for details on the first paper.

Reference

Yam, Aileen L. (1999), "Automating the Production of Customized PowerPoint Presentation Graphs by Integrating the Functionality of SAS Software with Microsoft Visual Basic for Applications", *Proceedings of the Twenty-Fourth Annual SAS Users Group International Conference*, 24, 141-145.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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

For additional information, contact:

***Aileen L. Yam
PharmaNet, Inc.
504 Carnegie Center
Princeton, NJ 08540-6242***
