Paper 232-25

GenHTML: A Library for Converting SAS® Datasets to HTML Tables

Robert Burnham, Amos Tuck School of Business, Dartmouth College, Hanover, NH

ABSTRACT

GenHTML is a library of macro functions that allow users to easily generate HTML tables from SAS® datasets. The library is ideal for producing static pages, but is also fast enough to produce dynamic pages when called from a CGI script. The library is freely available at http://www.dartmouth.edu/~bburnham and is distributed with the hope that users will continue development of the library and share their contributions.

INTRODUCTION

GenHTML was developed to meet three requirements. First, the library had to produce HTML tables. This allows end users of our web site to retrieve their web queries either as tables or as Microsoft Excel® spreadsheets. Since Excel® can translate HTML tables and maintain their formatting, we were able to implement the option of directing queries to either Excel® or web browsers by changing only one line of code in the HTTP header. The second requirement was that the routines needed to preserve as much of the format of the SAS® data as possible. For example, the library looks at whether the contents of a cell should be left or right justified depending on the format of the variable in the dataset. The final design goal was to allow users to set constants for certain formatting values (e.g. the color of a table or the width of the table's border) so that all tables generated from the library adhered to a defined style.

The library was developed using SAS® 6.12 and does not require any products beyond BASE SAS®. Beginning SAS programmers should be able to effectively use this library in their own programs. Programmers interested in modifying and extending the library should have a basic understanding of how the SAS® macro facility works and be familiar with HTML table syntax.

GENHTML: A SAS® TO HTML LIBRARY

The library is currently composed of twenty-two macros with extensive comments and documentation. While this collection of macros serves as a useful toolbox for developing more complicated procedures, a complete HTML page can be generated using four basic procedures:

- %INIT_FH opens a new HTML file for output
- %B_HTML opens an HTML tag
- %REP2HTML generates the HTML table code
- %E_HTML closes the HTML tag

The macro library works by running PROC CONTENTS on a dataset to determine the type, format and alignment of the variables in a dataset. The macros use the results to generate an HTML table that maintains these formats and alignments.

SIMPLE EXAMPLE

This example takes some sales data from several stores and displays it as an HTML table. Imagine that the data was retrieved from the data warehouse and that PROC REPORT was used to produce a report dataset called "salesrep." The code might look like the following:

```
proc report data=sales out=salesrep;
  column storeid date customer cost;
  define storeid / group;
  define date / group;
  define customer / group;
  define cost / sum;
  break after storeid / skip ol summarize;
  rbreak after / skip dol summarize;
  run;
```

The output dataset contains the variables storeid, date, customer, and cost which contain the store id, date of purchase, customer name, and total cost respectively. To produce an HTML table of this report, we could use the following code:

```
FILENAME htmlFile "output.htm";
%INIT_FH(htmlFile);
%B_HTML(htmlFile, Store Sales Reports);
%LET title=Totals by Store ID;
%LET vars=storeid date customer cost;
%REP2HTML(htmlFile, salesrep, layout,
  &vars, &title);
%E_HTML(htmlFile);
```

The first line defines the SAS file reference, htmlFile, as the file "output.htm." This file reference will be passed to all of the macro library calls so that the output will be directed to the proper file. This also allows the user to direct the output of the library functions to multiple HTML files as needed.

The second line (%INIT_FH) initializes our output file, referred to as "htmlFile", by simply passing the reference to a SAS FILE statement. The FILE statement defaults to replacing the previous contents of the file.

The %B_HTML command begins writing HTML to the output file by writing the HTML header tags and by setting the value of the second macro argument as the title of the web page. This is not the title of the table, but rather the HTML page title as indicated in title bar of the web browser.

The next two %LET commands define two additional parameters that are going to be passed to %REP2HTML. The &title macro variable contains the heading for our HTML table and &vars contains the names of all of the variables that are going to be included in the table. The library's %SPLIT function will automatically parse the contents of &vars into an array of string variables when they are needed.

The %REP2HTML function generates the HTML code for our table. %REP2HTML is not a single function, but a collection of calls to other members of the library that are responsible for different aspects of converting the data to HTML. %REP2HTML was written as an example of the type of HTML functions that can be implemented using the library.

The one variable that is passed to %REP2HTML that has not been described is &layout. This variable will be used as the name of a dataset that the library will use to hold the results of the call to PROC CONTENTS.

The HTML page is completed by call to %E_HTML that closes the open BODY and HTML tags.

BEHIND THE SCENES

This library was extremely easy to develop with the SAS® language, which has a great variety of functions for dealing with character strings. However, there were several challenges that I needed to solve in order to get everything working correctly.

In order to make the library easier to use, I wanted all of the functions to accept lists of variables as strings. This allows users to pass parameters to the library functions in the same way that they would in the VAR statement of a SAS® PROC. To accomplish this, I needed a function to split a delimited string and place the results into an array. An example of this functionality is found in the standard UNIX bufsplit() function or in Perl's split() function. To get this functionality, I implemented my own %SPLIT function:

```
/*
 %split(delimit, string, retArr, maxArr)

 This is a generic utility function for
 splitting a string on a single character
 delimiter and placing the results into an
 array (&retArr) with a maximum size of
 &maxArr.  A count of the total number of
 elements in the array is returned in
 &maxArr.

*/

%macro split(delimit, string,
         retArr, maxArr);
  copy = &string;
  arrCount = 0;
  do _I_ = 1 to &maxArr;
    l = indexc(copy, &delimit);
    if (l > 1) then do;
      arrCount + 1;
      &retArr{_I_} =
     substr(copy, 1, (l - 1));
      copy =
     trim(left(substr(copy, l+1)));
    end;
    else &retArr{_I_} = "";
  end;
  &maxArr = arrCount;
%mend;
```

The second challenge was dealing with timing issues related to making macro calls using CALL EXECUTE. SAS® code generated by an invocation of CALL EXECUTE in a DATA step does not get executed until after the DATA step is complete. As a result, it took some planning to insure that the library's DATA step and CALL EXECUTE generated code wrote the HTML code in the correct order.

The GenHTML library handles these timing issues by having two copies of some of the HTML functions. For example, there are two functions, %B_TR and %SAS_B_TR that each produce the opening <TR> tag for beginning a new table row in HTML. In the library, %B_TR is implemented so that it can be executed immediately, while %SAS_B_TR actually writes out the SAS® code (several PUT statements) to a temporary file that is executed (using an %INCLUDE) statement and then deleted when processing is complete.

CONCLUSION

GenHTML is a collection of macros that make generating HTML tables from SAS® data sets extremely easy. The library produces attractive tables in good HTML syntax and is easy to extend and modify for custom reporting formats. HTML tables are a very useful data exchange format that can be read by many different applications, including Microsoft Excel®. As a result, many users will find the library useful as useful in a non-web environment as they do in developing web based applications.

REFERENCES

SAS Institute Inc., *SAS® Macro Language: Reference, First Edition*, Cary, NC: SAS Institute Inc., 1997.

Stephen Spainhour and Valerie Quercia, *WebMaster in a Nutshell*, Sebastopol, CA: O'Reilly & Associates, Inc., 1996.

Larry Wall, Tom Christiansen and Randal L. Schwartz, *Programming Perl, Second Edition*, Sebastopol, CA: O'Reilly & Associates, Inc., 1996.

David A. Curry, *UNIX Systems Programming for SVR4*, Sebastopol, CA: O'Reilly & Associates, Inc., 1996.

Lincoln Stein, *Official Guide to Programming with Cgi.Pm*, New York, NY: John Wiley & Sons, 1998.

ACKNOWLEDGMENTS

Developing this library would not have been possible without the support and continuous encouragement of Tuck Research Computing and the Amos Tuck School of Business Administration at Dartmouth College.

I would like to thank Kathryn Sabadosa and Bill Marble for their comments and suggestions. I would also like to thank the members of SAS-L that have so generously shared their wisdom with me.

CONTACT INFORMATION

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

Robert Burnham
Amos Tuck School of Business
100 Tuck Hall
Dartmouth College
Hanover, NH 03755
Work Phone: (603) 646-2518
Email: bburnham@dartmouth.edu
Web: www.dartmouth.edu/~bburnham