P-310

Static SAS (R) Web Publishing (SAS/IntrNet @ Social Work)

Roderick Rose, Jordan Institute for Families, UNC-Chapel Hill

ABSTRACT

This course is for experienced SAS programmers who want to learn static SAS web publishing using basic SAS commands, such as data statements and macros, and also with the Output Delivery System (ODS). Taking this class requires no previous HTML programming skills, nor does it require experience with ODS. It is assumed that the user has SAS programming experience and some familiarity with Windows 95. During the class, the student will download programs and data sets and "learn-by-doing" by authoring a web page, using FTP and building SAS programs. This class serves primarily as an overview so as to familiarize students with the options available. Further exploration of one or more web publishing options in SAS will require turning to other sources.

The Web Formatting Tools and ODS are very simple to use to convert your existing libraries of SAS programs, such that they can produce static web output. The advantages of static SAS web publishing are numerous. Your reports can be distributed worldwide for a cost, at the margin, of next-to-nothing. They can be updated periodically and re-distributed, also at a cost of next-to-nothing. The means of distributing and displaying the information has been relegated to a common software tool - the web browser. In the future, users may bypass traditional information channels and go directly to the web page instead, leaving you and your associates free to be more productive.

THE CLIENT/SERVER APPLICATION

The Client/Server Application consists of several computers, protocols, and files, most of which are invisible to users of the Internet. Understanding the client/server application does not require a great leap for someone with basic Internet skills. It will help you understand your role in the process, and give you an idea of how any random computer user retrieves the information you've put online.

Web pages must be stored on a web server, which is a computer connected to the Internet. The files on the web server are accessible by users worldwide. Each server is assigned a unique numeric identifier called an IP address, making it accessible to others. The domain, which you recognize as www.unc.edu or www.cnn.com, for instance, is translated into the IP address by routers, computers which control the flow of traffic on the Internet.

The programmer must:

- Determine what content will be made available on your static pages
- Write the SAS programs that generate the static content
- Debug the content for errors in both SAS and HTML
- Make the material available on a server by uploading it

The web pages are retrieved over the Internet by a user whose computer is referred to as the "client". The user types the domain into the address bar (or selects a link) and the client sends a request to the corresponding server. The server responds with the requested information. In many cases, the requested information would be a static web page. In order to retrieve this information, the client must have appropriate access to the server. This would mean a computer with an Internet connection and a web browser, like Netscape Navigator or Microsoft Internet Explorer.

The audience will:

- Use a computer (the client) to find your website
- View the pages you have already designed and uploaded

It is not the job of the audience to determine what content will be made available on the pages. That is strictly relegated to a dynamic application. With static, the audience views exactly what you have created beforehand.

HTML

HTML is the predominant "language" of the web. You will need to know a little in order to use DATA _NULL_ to generate HTML on your pages.

Elements & Attributes
Everything on an HTML document is an element. There are three different types distinguished by their use - structural elements, object elements, and formatting elements. Many elements can be formatted to your liking by employing "attributes". Common attributes include color and size options.

Tags
Each element must be invoked by a "tag". Every element on an HTML-formatted page, with the noted exception of readable text itself, must be enclosed within "<" and ">".

Each of these elements requires a "start tag" to invoke the element, ie.,<b> **to tell the browser to format everything that follows with a bold face. Additionally, certain elements require "end tags" to tell the browser to stop formatting the element. The end tag is indicated by a "/" inserted inside the tag, ie.,** </b> , which will tell the browser to return to whatever formatting options prevailed before the element was invoked with the start tag.

Rules
There are no strict rules of organization to recall when writing a simple HTML document. However, you should note:

1. As you might expect, the browser parses the page from top to bottom. This becomes very important when fixing your mistakes. If there is a mistake, the browser will show whatever it understands and mis-format everything that comes after.
2. No matter how much white space you insert between one letter and the next, it will format only one white space.
3. Following directly from (2), you can put your code on as many lines as you like, or on as few as one.
4. Almost nothing on the page is case-sensitive (text is one obvious exception).

We use the four-character extension file-naming convention, ".html", but you can just as easily use ".htm".

WRITING HTML WITH DATA _NULL_

To produce HTML code as SAS output, you will employ DATA _NULL_, using the PUT statements to encapsulate the HTML code from the previous lesson.

The key feature to keep in mind is that instead of directly authoring an HTML document yourself, you are directing SAS to author one for you. You must tell it to write exactly what code you need in order to produce the desired output. HTML code can be generated in a DATA _NULL_ step using FILE and PUT statements:

```
DATA _NULL_;
FILE 'the file name (full path or
relative) to write to,
    with .html file extension';
PUT 'All desired HTML Elements';
PUT 'More HTML';
...RUN;
```

Since no SAS data set is being created, we specify the _NULL_ keyword in the DATA statement. You can use more than one DATA _NULL_ in a SAS program to create as many .html files as you want, or to append to one .html file. You can also use as many PUT statements as are necessary to generate the desired HTML output.

STEPS
The process of developing static SAS-based output on the web involves several steps: HTML, SAS programming, uploading and debugging.

Step 1: HTML
Use what you learned in the previous lesson to write a page of HTML, like the following:

```
<html><title>A Sample Data Set by DATA
_NULL_</title>
<body>
<P class="h1">A Sample Data Set</p>
<ol>
<p>
<a
href="http://help.unc.edu/statistical/sas
intrnet/staticnotes_4.html">
Go Back</a>
</OL>
</body>
</html>
```

Step 2: SAS Programming
The best way to proceed is to program your SAS code around what you've already developed using DATA _NULL_, FILE and PUT.

```
/* Example of SAS Output to HTML */
/* html1.sas */
data _null_;
file 'html1.html';
put '<html>';
put '<title>
A Sample Data Set by DATA
_NULL_</title>';
put '<body>';
put '<P class="h1">A Sample Data
Set</p>';
put '<ol>';
put '<p>';
put '<a
href="http://help.unc.edu/statistical/';
put 'sasintrnet/staticnotes_4.html">Go
Back</a>';
put '</OL>';  put '</body>';
put '</html>';
run;
```

The code follows the protocol of any SAS program, including the use of a semicolon at the end of a line of code and comments between /* and */. The file name you assign to your output file must have the extension ".html" or ".htm". The elements between the quotes, in the put statements, are the exact HTML tags that will be used to format a .html file.

Step 3: FTP
The last lesson in this part of the presentation is to place your material online. You will become familiar with several methods of file transfer protocol. For now, we will skip this step. It is not necessary to upload static files to the server before debugging them, as it can be done "off-line" usually much quicker.

Step 4: Debugging
HTML and DATA _NULL_ are from two different systems, and each has their own problems. Together, they can cause even greater problems. You have to be especially careful regarding unbalanced quotes.

To review problems, you should refer to both the log from the SAS program as well as the page you've created. To avoid compounding problems, I encourage you to write an HTML file first, formatting it and working the "kinks" out. Then build PUT statements in standard SAS syntax around the HTML code and run the program in SAS.

PUTTING YOUR FILES ON THE WEB

The last step in this process is to actually place the files you have created on a server which is accessible worldwide. In this class we strictly use the FTP Access Method, and we encourage you to use it as well.

Drawbacks

- When the program runs, the user will be prompted to enter the password for the user account to which the output is being sent. A prompt will occur for each file, and for each time the file is edited and submitted. This is for security purposes and can be bypassed.
- Only files created by SAS can be uploaded this way. Other static files, like logo or header images, must be uploaded using one of the other FTP clients.

Invoking and Uploading in One Step
To use this simple tool, a few statements must be added to your SAS program:

```
filename fileref ftp 'filename'
cd='~userID/public_html'
host='hostname'
```

Finally, specify the userID again, and 'prompt', such that the program prompts you for the password:

```
user='userID' prompt;
```

As an alternative to prompt, you can use pass=your_isis_password. This is not recommended because it makes your login password clear to anyone with access to the program. But it does save you the trouble of having to constantly re-type your password.

There should be a semi-colon on the end. It should come before you specify any output destinations (using DATA _NULL_, macros or ODS). You can copy these exact statements and simply insert your specific references as required.

THE DATA SET FORMATTER - %DS2HTM

The Data Set Formatter converts SAS data sets to well-defined HTML tables. %DS2HTM is a macro which must be invoked in your SAS program and cannot be called from a SAS command line. A list of common arguments, required and optional, are available below. More are available at the SAS Institute website.

The Data Set Formatter is invoked only once every time you want to generate output (as opposed to twice, which as you will see shortly, is how the Tabulate and Output formatters work).

Use
The Data Set Formatter can replace PROC PRINT as an output procedure. For instance, the following represents the print procedure from a working SAS program:

```
proc print data=out.ds1;
where trancode = 8 and alltment > 600;
var alltment trancode certbeg certend;
by certbeg;
run;
```

Alternatively we can use %ds2htm to generate the same output in HTML form:

```
%ds2htm(data = out.ds1,
htmlfref = ds2,
var = alltment trancode certbeg certend,
by = certbeg,
center = y,
obsnum = y,
where=(trancode eq 8 and alltment gt
600),
openmode=replace,
runmode = b);
run;
```

OUTPUT FORMATTER - %OUT2HTM

The HTML Output Formatter macro is utilized to present data output in the format of an HTML document.

Use
Write a SAS program as you normally would. Include the following in your program, at the location where the data is to be processed, starting with the macro call "%out2htm":

```
%out2htm(capture=on,
window=output,
runmode=b);
```

CAPTURE=ON tells SAS to output HTML. After specifying to SAS that it should capture output using %OUT2HTM, you specify the procedures it should capture:

```
proc print data=out.ds1;
where trancode = 8 and alltment > 600;
var alltment trancode certbeg certend;
by certbeg;
run;
```

Each CAPTURE=ON is accompanied by a CAPTURE=OFF, after all procedures to send to web output have been specified:

```
%out2htm(htmlfref=out2,
capture=off,
window=output,
openmode=replace,
tcolor=red,
runmode=b);
```

STATIC OUTPUT USING ODS HTML

ODS HTML is a versatile alternative to using the HTML Formatting Macros. It provides more options and better default formatting. ODS HTML can be used with any of the HTML Formatting Macros and Data _NULL_. You can place as many procedures on one .html page as you would like, and they can all be processed while the HTML destination is opened.

The instruction of ODS HTML skills is centered around building a SAS program from top to bottom, much in the same way that I did with the HTML Formatting Tools.

Opening the HTML Destination
The files specified (in this case a body page, a contents page and a frame page in which to display them) can be named in any order. The options refer to each file and follow the file specification in parenthesis. If more than one option is specified, they are all in the same parentheses, separated by commas. Remember that it is a good idea to use a fileref, with the FTP Access Method, in which case your file specifications would refer to filerefs. If instead you choose to name files you need to contain the file names within quotes.

```
ODS HTML
file = ods2 (NO_TOP_MATTER)
contents= cods2 (NO_TOP_MATTER)
frame=fods2;
```

Directly after the last specification, you need to place a semicolon (as shown after frame). There are additional file specifications which you can name in your program as well (they should be placed before the semicolon):

```
path='/afs/isis.unc.edu/r/a/rarose/public
_html (URL='http://www.unc.edu/~rarose')'
headtext='<title>Document Title
Name</title>'
anchor='abbreviation'
newfile = NONE | OUTPUT | PAGE | PROC
```

Producing Output
At this point you would specify the procedures you would like SAS to perform. The syntax follows standard SAS procedural syntax. The following example assumes that I have specified a libname out and that a data set ods1 exists.

```
proc contents data=out.ods1;
run;

proc print data=out.ods1;
where trancode = 8 and alltment > 600;
var alltment trancode certbeg certend;
by certbeg;
run;
```

ODS will generate as many procedures on one page of HTML code as you specify. In this case, it will output the contents and print procedures. In order to finish the program you need only close the destination.

Closing the HTML Destination
In order to close the HTML destination, use ODS HTML CLOSE:

```
ODS HTML CLOSE;
```

SUMMARIZING KEY CONCEPTS

We provided an introduction to several useful tools:
- HTML
- Uploading with the FTP Access Engine
- HTML output with DATA _NULL_
- HTML Formatting Macros
- ODS

These tools, used together or separate, provide great versatility:
- You can use any combination of these tools to put static SAS output online.
- You can write several SAS programs to append to one output file by specifying the append, mod or no_bottom_matter/no_top_matter options, depending on the tool you are using.

- Static output skills are useful when programming dynamic online content.

You should be reminded of key hints that will simplify your work:
- If you intend to write HTML output directly by using a Data Step, write the HTML beforehand, view it, and then build the SAS program around it.
- Use the programs provided on this website around which to build your own programs.
- Return here in the future and delve deeper into each tool's programming features by reviewing other pages on this website.
- Use the SAS Institute's website to supplement our instruction.

And finally, for the truly overwhelmed: recognize that, out of necessity, you need one of each of the following:
- A method of sending output to a web page (The HTML Formatting Macros, ODS).
- A method of adding headers, footers and other complementary elements to the page (The Data Step, ODS)
- A method of putting your work on the server for access (by working in your AFS space, or by FTP uploading, either by client, or by SAS FTP Access Method)

CONTACT INFORMATION

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

Roderick A. Rose
Jordan Institute for Families
CB 3550
301 Pittsboro St.
Chapel Hill, NC 27599
Email: rarose@email.unc.edu