Paper 103-2007

Simple and Effective: Using ODS Destinations to Facilitate Data Management

Paul Gorrell, Social & Scientific Systems, Inc., Silver Spring, MD

ABSTRACT

A basic component of the management of project data and metadata is easy and convenient access to the information you (and your clients) need. The approach outlined here has two general themes: (i) the various ODS output destinations allow you to choose the application that best fits you and your clients' needs for readily accessing information; (ii) with only minimal additions to the skills you already posses as a SAS® programmer, you can significantly enhance the usability and accessibility of data and documentation. Although *usability* is a term most often associated with the development of Web pages, the basic principles are applicable to data management. Information about project data and programs should be easily located and accessed.

The general approach will be illustrated by looking at the navigational and display properties of three ODS output destinations: HTML, PDF and Excel (via the ExcelXP tagset). The HTML destination allows for easy navigation among metadata files. The PDF destination is well-suited for final project documentation and multi-platform compatibility. Excel, especially the use of the Autofilter option, allows for easy navigation within a file or table. The techniques discussed in this talk require knowledge of only basic SAS programming statements and procedures, but use ODS destinations to expand the usefulness of the SAS data and metadata you're already familiar with. The result of applying these simple techniques is a set of inter-related files that allow you to access a wide array of important project information with a minimal number of mouse clicks.

INTRODUCTION

The Data Management Association (DAMA) defines data management as, "... the development and execution of architectures, policies, practices and procedures that properly manage the full data lifecycle needs of an enterprise." This paper will focus on access to, and navigational properties of, project data and metadata. There's an ad for a document management system that says, "It's not what you don't know that can hurt you. It's what you can't find." But it's not only what can't be found, but also what busy programmers don't bother to look for because the search is annoying or time consuming or has proved unproductive in the past. A basic property of any approach to data management and availability is usability. A common definition of usability is

> The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of user. (ISO 9241-11)

The Federal government provides a good resource for usability information at www.usability.gov. Here are two important, but unsurprising, facts, based on studies of Web site visits:

- Users are impatient
- Users are usually under some form of time pressure

These points, of course, apply not just to Web sites, but to any application or system that people use to access information. Systems for the management and documentation of project information are no exception. It is not uncommon in the workplace for programmers to create their own personal collection of sticky notes, cheat sheets and lists of assorted types for keeping track of data. Therefore it's important to design methods for accessing project data that actually get used in the real world of time pressure and competing priorities. According to Usability.gov, three important factors are:

- Ease of learning
- Efficiency of use
- Memorability

Ease of learning has two dimensions. The first is in the creation of the system for accessing project data. As we'll see this is accomplished by taking advantage of the basic skills most SAS programmers already have. The second is the use of the system once in place. This is accomplished by using the navigational functionality built into HTML,

PDF and Excel files. This also allows for efficiency of use. Generally a system easy to learn is one that is easy to use. Memorability is accomplished by reducing the number of distinct facts you need to remember to access information in different files. The links between and within files not only reduce the number of mouse clicks need to get from File A to File B, but they also obviate the need to remember the location and names of the different files that are the destinations of the links.

For data management, the importance of the ability within SAS to choose different output destinations is that it allows programmers and managers to consider usability criteria in designing systems for accessing data and metadata. In the following sections we'll describe the specific advantages of HTML, PDF and Excel output, but the general theme will be reducing the amount of background knowledge needed to access the varied files which contain project information. For the most part a single entry file can be determined and then links to associated information can be created within these files. The intelligent placement of links to associated files makes for a user-friendly set of inter-connected files.

The definition of usability refers to "specified users". The focus of this paper is an approach to data management and accessibility that can be both created and used by SAS programmers. But the use of varied ODS destinations allows for an expansion of the "specified users" to include Project Managers, others in your organization, and clients who may not use SAS but are certainly familiar with Excel and PDF files. One of the basic "needs of an enterprise" is that project information be readily located and accessed not only during the lifecycle of the project, but quite often down the road as well.

ODS DESTINATIONS AND SYNTAX

With ODS destinations you choose the type of output destination that best fits your needs. In this paper we'll discuss generating HTML, PDF AND Excel output (see Haworth's book on ODS Basics for an extensive discussion of ODS destinations). What's especially nice about ODS is that you can apply your SAS programming skills to creating different types of output. Of course the more you know about the formal properties of your intended output destination the better but, in general, learning just a little of the necessary syntax and vocabulary goes a long way. We are going to keep it very simple in this paper and focus on the minimum needed to direct output to a particular destination.

```
ODS destination FILE='path\filename';

   … SAS code that produces output…

ODS destination CLOSE;
```

For *destination* we will substitute either "HTML", "PDF", or "TAGSETS.EXCELXP". In the following sections we will, in turn, look at these types of output destinations with respect to their navigational and data-access properties. We will use specific code and screen-shot examples to give you a concrete understanding of these SAS programs and the resulting output.

HTML OUTPUT

HTML output isn't just for Web sites. You can take advantage of its navigational properties to create a set of connected files that are accessible to anyone with a browser. Further, generating HTML output, and adding links where you need them to facilitate rapid data access, requires only minor enhancements to the type of SAS programming you're already familiar with. For example, here's the HTML code used in a SAS program for turning text into a hyperlink, where TEXT is whatever text you want to use in TITLE, FOOTNOTE or PROC FORMAT VALUE statements. The <A> anchor tag is used to create a link to another document. The *href* attribute is used to specify which file to link to.

```
<A href='path\filename'> TEXT </>
```

We can use this in a SAS program as follows:

```
TITLE  "LIST OF DCS DATA SETS";
TITLE2 "<A href="C:\DOC\DCS2003VAR.HTML'> CLICK FOR VARIABLE LIST </A";
```

Here "CLICK FOR VARIABLE LIST" is a hyperlink to the indicated HTML file. Note that, aside from the HTML code, it's just an ordinary TITLE statement.

Before proceeding let's first take a look at the HTML file in Figure 1.

Figure 1

LIST OF DCS DATA SETS
CLICK FOR VARIABLE LIST

DATA_SET	TOPIC	PROGRAMMER	ORIG_DATE	NUM_OBS	NUM_VARS
DCS2003V1	Diabetes	Paul Gorrell	25JUL03:00:32:06	1529	25
DCS2003V2	Diabetes	John Smith	17NOV03:18:18:46	1403	29
DCS2003V3	Diabetes	Mary Stuart	06JUL04:05:52:06	1434	33
DCS2003V4	Diabetes	Paul Gorrell	22FEB05:17:25:26	1434	45
DCS2003V5	Diabetes	Susan Jones	12OCT05:04:58:46	34215	52
DCS2003V6	Diabetes	Susan Jones	09DEC05:01:52:07	34215	57
DCS2003V7	Diabetes	John Smith	16FEB06:12:32:09	34215	57
DCS2003V8	Diabetes	Mike Mitchell	23MAR06:05:52:10	34215	61
DCS2003V9	Diabetes	Paul Gorrell	31MAY06:16:32:13	34215	62

This file shows all the data sets created by a small project. It also shows the topic (the same here for all data sets), the programmer, the date the data set was created, as well as the number of records and variables. There's more information that could be included here, but this serves as a useful example. Consider a situation where this project was completed a year or so ago and you're about to have a meeting where a new phase of the work is being considered. Having the information shown in Figure 1 is incredibly useful. Consider further that you can display this at the meeting and easily navigate to information about specific data sets or programmers.

This navigation would be easy because each underlined word or phrase is a hyperlink. In the TITLE2 statement above we've already seen how to create the hyperlink CLICK FOR VARIABLE LIST. From the *href* specification we know that this is a link to a file named C:\DOC\DCS2003VAR.HTML. But the file in Figure 1 has other links as well. Each value of the DATA_SET variable and one value of the PROGRAMMER variable are hyperlinks. Let's take a step back now so we can understand where the file DCS2003VAR.HTML comes from, and how it relates to other files we'll discuss.

As an example we'll make up a small project that created a set of nine SAS data sets named sequentially from DCS2003V1 to DCS2003V9. We're going to use a bit of PROC SQL to get some metadata from the SAS-generated dictionary tables associated with these data sets (for a paper with a lot of useful information about SAS dictionary tables and views, see DiIorio and Abolafia's SUGI 29 paper; see also Davis' SUGI 26 paper on dictionary views). The examples below, as DiIorio and Abolafia's paper shows, are but the tip of the iceberg of all the useful information in these tables and views.

These nine data sets are in a directory with the LIBREF of DCS. To create a data set with the metadata we want, we use the following code:

```
PROC SQL;
 CREATE TABLE MDATA.DCS2003_T AS
 SELECT MEMNAME AS DATA_SET,
        CRDATE AS ORIG_DATE,
        NOBS AS NUM_OBS,
        NVAR AS NUM_VARS
```

```
 FROM DICTIONARY.TABLES
 WHERE LIBNAME = 'DCS';
QUIT;
```

We're using the keyword AS to rename the variables here to make them a bit more informative. From DICTIONARY.TABLES we've now extracted the following information for each of these data sets:

- data set name (DATA_SET)
- creation date (ORIG_DATE)
- number of observations (NUM_OBS)
- number of variables (NUM_VARS)

This information is now in the data set MDATA.DCS2003_T (where "_T" is a mnemonic for table (or, data-set) information). Note that we haven't yet explained the source of the TOPIC and PROGRAMMER variables. We'll return to this. Just as we created a data set with table-level information, we can create a data set with information about the variables in the data sets.

```
PROC SQL;
   CREATE TABLE MDATA.DCS2003_C AS
   SELECT MEMNAME AS DATA_SET,
          NAME AS VAR_NAME,
          TYPE AS VAR_TYPE,
          LABEL AS LABEL,
          FORMAT AS FORMAT
   FROM DICTIONARY.COLUMNS
   WHERE LIBNAME = 'DCS';
QUIT;
```

Note that here we're using DICTIONARY.*COLUMNS*, so we are getting information about columns, i.e. variables.

Once we've created these data sets, it's simple to create HTML versions by using ODS. The following code creates an HTML file from the DCS2003_T data set. Putting aside for the moment the TOPIC and PROGRAMMER variables, this code will generate the file shown in Figure 1.

```
ODS HTML FILE='C:\DOC\DCS2003.HTML';

TITLE 'LIST OF DCS DATA SETS';
TITLE2 "<A href="C:\DOC\DCS2003VAR.HTML'> CLICK FOR VARIABLE LIST </A";

PROC PRINT DATA= MDATA.DCS2003_T NOOBS;
   FORMAT DATA_SET $DSNF. PROGRAMMER PROGF. ;
RUN;

ODS HTML CLOSE;
```

Except for the syntax and symbols for HTML tags, we're only using the simplest aspects of SAS programming: TITLE statements, PROC PRINT, and two ODS statements. We'll come back to the FORMAT statement, but now let's use similar code to create an HTML file from the DCS23003_C data set. Recall that this is the data set with information about variables. First we'll sort the data set so the HTML file has the metadata organized the way we want it. Of course the variables you extract from the SAS-generated metadata, and how you sort or organize them will be dependent on the particular needs of your projects, programmers and clients.

```
PROC SORT DATA= MDATA.DCS2003_C;
   BY VAR_NAME DATA_SET;
RUN;

ODS HTML FILE='C:\DOC\DCS2003VAR.HTML';

TITLE 'DCS VARIABLES AND DATA SETS';
FOOTNOTE "<A href="C:\DOC\DCS2003.HTML'> RETURN TO DATA SET LIST </A";
```

```
PROC PRINT DATA= MDATA.DCS2003_C NOOBS;
   VAR VAR_NAME DATA_SET;
   FORMAT DATA_SET $DSNF. ;
RUN;

ODS HTML CLOSE;
```

The output file DCS2003VAR.HTML is the one referenced in the TITLE2 statement above (the destination for the hyperlink CLICK FOR VARIABLE LIST. This file is illustrated in Figure 2 (only selected rows from this data set are shown). The FOOTNOTE statement illustrates that once you've learned how to insert hyperlinks in one context (e.g. TITLE statements), it's easy to insert them elsewhere. Later on we'll see them used with PROC FORMAT.

Figure 2

DCS VARIABLES AND DATA SETS

VAR_NAME	DATA_SET
AGECAT5C	DCS2003V1
AGECAT5C	DCS2003V2
AGECAT5C	DCS2003V3
AGECAT5C	DCS2003V4
AGECAT5C	DCS2003V5
AGECAT5C	DCS2003V6
AGECAT5C	DCS2003V7
AGECAT5C	DCS2003V8
AGECAT5C	DCS2003V9
ALL3TSTS	DCS2003V5
ALL3TSTS	DCS2003V6
ALL3TSTS	DCS2003V7
ALL3TSTS	DCS2003V8
ALL3TSTS	DCS2003V9
STRKD	DCS2003V6
STRKD	DCS2003V7
STRKD	DCS2003V8
STRKD	DCS2003V9

RETURN TO DATA SET LIST

This type of file is used to give a quick overview of which variables are in which data sets. As we'll see below, if you click on the links for specific data sets, you get more information about the variables. This file also shows, at a glance, where in the sequence from DCS2003V1 to DCS2003V9 a variable was created. For example, the variable AGECAT5C was in DCS2003V1 and kept in all subsequent data sets. But ALL3TSTS wasn't created until DCS2003V5.

Let's take a look now at the code that creates the HTML files with information specific to each data set listed under DATA_SET in Figures 1 and 2. The output is shown in Figure 3.

```
ODS HTML FILE= 'C:\DOC\DCS2003V1.HTML';

TITLE 'DCS2003V1';
TITLE2 "<A href="C:\DOC\DCS2003VAR.HTML'> CLICK FOR VARIABLE LIST </A";
TITLE3 "<A href="C:\DOC\V1LOGLST.HTML'> CLICK FOR LOG&LST FILE </A";

FOOTNOTE "<A href="C:\DOC\DCS2003.HTML'> RETURN TO DATA SET LIST </A";

PROC PRINT DATA= MDATA.DCS2003_C;
   VAR VAR_NAME VAR_TYPE LABEL FORMAT;
   WHERE DATA_SET = 'DCS2003V1';
RUN;

ODS HTML CLOSE;
```

The two TITLE statements and the FOOTNOTE statement each contain hyperlinks, so a display of this HTML file will also allow for one-click navigation to three other sources of data and metadata.

Figure 3

DCS2003V1
CLICK FOR VARIABLE LIST
CLICK FOR LOG&LIST FILE

VAR_NAME	VAR_TYPE	LABEL
ID	char	PERSON ID
SEX	num	SEX
DIABDX	num	DIABETES DIAGNOSIS
DSDIA	num	DIABETES DIAGNOSIS BY HEALTH PROF
DSA1C	num	TIMES TESTED FOR A-ONE-C
DSCKFT	num	TIMES FEET CHECKED FOR SORES
DSEY04	num	DILATED EYE EXAM IN 2004
DSEY03	num	DILATED EYE EXAM IN 2003
DSEY02	num	DILATED EYE EXAM IN 2002
DSEB02	num	DILATED EYE EXAM BEFORE 2002
DSEYNV	num	NEVER HAD DILATED EYE EXAM

VAR_NAME	VAR_TYPE	LABEL
DSKIDN	num	HAS DIABETES CAUSED KIDNEY PROBLEMS
DSEYPR	num	HAS DIABETES CAUSED EYE PROBS
DSDIET	num	TREAT DIABETES W/DIET MODIFICATION
DSMED	num	TREAT DIABETES W/MEDS BY MOUTH
DSINSU	num	TREAT DIABETES W/INSULIN INJECTIONS
AGECAT5C	num	

RETURN TO DATA SET LIST

With the HTML file in Figure 3 you can look at variable-level information for a particular data set (only a small sample of which is shown here). You can easily navigate to the project variable list shown in Figure 2 (CLICK FOR VARIABLE LIST) or to a copy of the LOG and LST files for the program that created the data set (CLICK FOR LOG&LST FILE). This can be particularly useful for accessing information in the program header or other LOG output. Any of a number of methods for converting text files to HTML can be used with SAS LOG files. In addition the footnote allows you to navigate back to the file listing all the project data sets. A good rule of thumb is to design the connections among files so that no file becomes a dead-end.

At this point we've seen how easy it is to use basic SAS programming skills, with a bit of HTML enhancement, to generate output that makes program output and project metadata easily accessible. We still have a few details to cover before moving on to PDF and Excel output. In Figure 1 we see two variables (TOPIC and PROGRAMMER) that we haven't shown a source for. Consider the following code:

```
DATA DCS.DCS2003V1 (LABEL= 'DIABETES/PAUL GORRELL');
   SET DCS2003V1;
RUN;

PROC SQL NOPRINT;
   CREATE TABLE DCS.DCS2003_T AS
   SELECT MEMNAME AS DATA_SET,
          PROPCASE(SCAN(MEMLABEL,1,'/')) AS TOPIC,
          PROPCASE(SCAN(MEMLABEL,2,'/')) AS PROGRAMMER,
          CRDATE AS ORIG_DATE,
          NOBS AS NUM_OBS,
          NVAR AS NUM_VARS
   FROM DICTIONARY.TABLES
   WHERE LIBNAME = 'DCS';
RUN;
```

As the first line of code shows, we're using the data set option LABEL= in order to include both a topic and programmer field in the information contained in DICTIONARY.TABLES. That is, the MEMLABEL variable contains the text information specified with the LABEL= option. Including this information is an easy requirement for project programmers. Further, since the text field can be 256 characters, you can potentially include quite a bit of information. This is especially true if you're clever and include a delimiter such as "/" so that you can later parse out different types of information.

This is what the SCAN function does in the PROC SQL block. In the original PROC SQL block we started with, we used the AS keyword to rename the variables. But we can also use functions on the left side of what now functions as an assignment statement. The SCAN function, in conjunction with AS, assigns the text preceding the slash to the variable TOPIC, and the text following the slash to the variable PROGRAMMER (the PROPCASE function just generates 'proper' case, i.e. lowercase for all letters except the initial letter of each word). This simple approach opens up lots of possibilities for tracking data set information and making it easily accessible at later points.

We've now seen how the HTML files are created and how the variables are extracted from DICTIONARY.TABLES and DICTIONARY.COLUMNS. But how are variable values created as hyperlinks? As we saw with the FOOTNOTE statement, we can use the HTML code we learned for TITLE statements in a number of other contexts. One of these is PROC FORMAT VALUE statements. By now this should look very familiar. We're taking some basic SAS code and adding an HTML enhancement.

```
PROC FORMAT;
   VALUE $DSNF
   'DCS2003V1' ="<A href='C:\DOC\DCS2003V1.HTML'> DCS2003V1 </A>"
   'DCS2003V2' ="<A href='C:\DOC\DCS2003V2.HTML'> DCS2003V2 </A>"
   'DCS2003V3' ="<A href='C:\DOC\DCS2003V3.HTML'> DCS2003V3 </A>"
   'DCS2003V4' ="<A href='C:\DOC\DCS2003V4.HTML'> DCS2003V4 </A>"
   'DCS2003V5' ="<A href='C:\DOC\DCS2003V5.HTML'> DCS2003V5 </A>"
   'DCS2003V6' ="<A href='C:\DOC\DCS2003V6.HTML'> DCS2003V6 </A>"
   'DCS2003V7' ="<A href='C:\DOC\DCS2003V7.HTML'> DCS2003V7 </A>"
   'DCS2003V8' ="<A href='C:\DOC\DCS2003V8.HTML'> DCS2003V8 </A>"
   'DCS2003V9' ="<A href='C:\DOC\DCS2003V9.HTML'> DCS2003V9 </A>";
   VALUE $PROGF
   'Paul Gorrell' ="<A href='C:DOC\PAUL GORRELL.HTM'> Paul Gorrell </A>";
RUN;
```

The effect of using format $DSNF is to associate the variable values on the left with hyperlink versions of the same text. The $PROGF format references a small HTML file with programmer-specific information. This could simply be contact information, or information specific to the project.

In this section we've seen the advantages of using the ODS HTML destination, in conjunction with adding hyperlinks to common SAS statements, to create metadata files for ongoing project data management and documentation. In the next section we take a look at PDF output and the ease of data access within output PDF files.

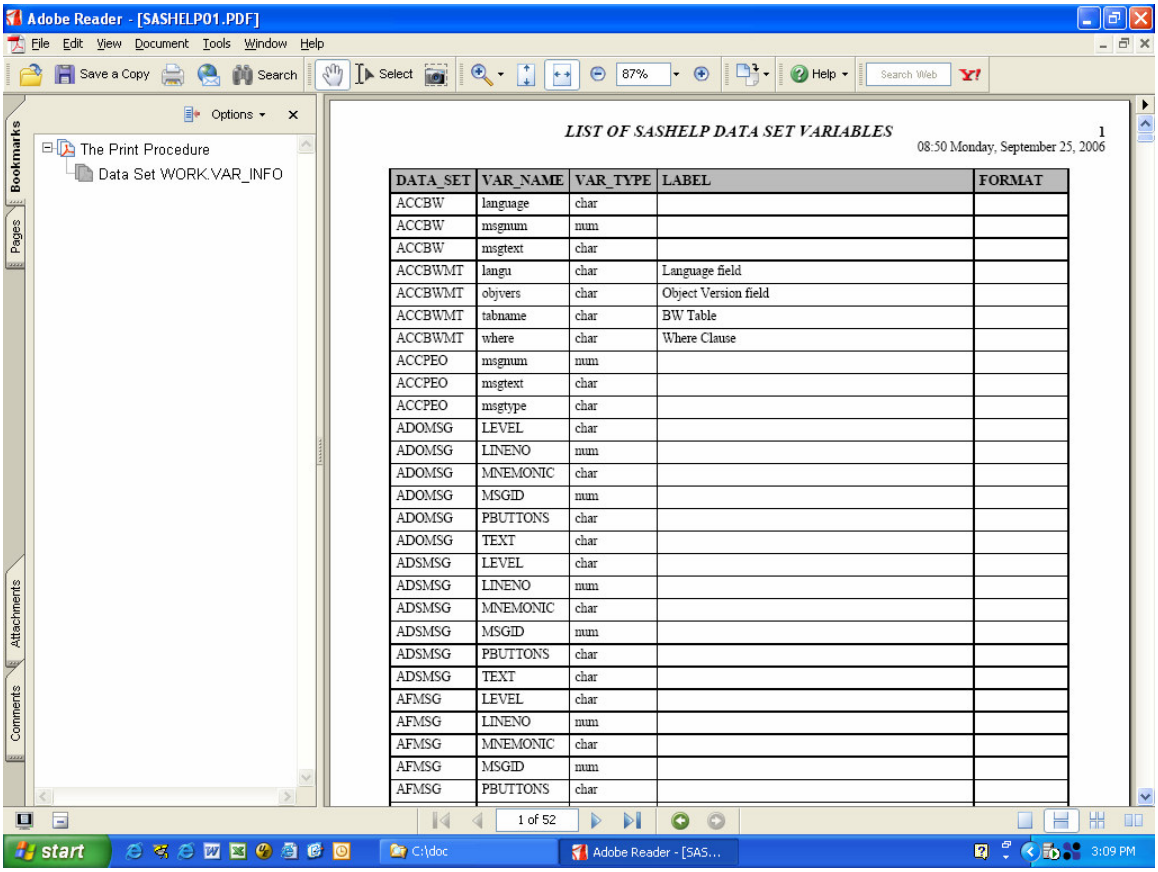
PDF OUTPUT

It's well known that PDF output provides good cross-platform compatibility, as well as 'what you see is what you print' advantages. But PDF output also provides useful within-file navigation. It is well-designed for project documentation that must be distributed to various clients and staff. In the following code we'll adapt the ODS statements we used for HTML output to generate PDF files. As we did for HTML, we'll also make use of some destination-specific enhancements.

```
PROC SQL NOPRINT;
   CREATE TABLE VAR_INFO AS
   SELECT MEMNAME AS DATA_SET,
          NAME AS VAR_NAME,
          TYPE AS VAR_TYPE,
          LABEL AS LABEL,
          FORMAT AS FORMAT
   FROM DICTIONARY.COLUMNS
   WHERE LIBNAME = 'SASHELP' AND MEMTYPE = 'DATA';
QUIT;

PROC SORT DATA= VAR_INFO;
   BY DATA_SET VAR_NAME;
RUN;

ODS PDF FILE='C:\DOC\SASHELP01.PDF';

TITLE 'LIST OF SASHELP DATA SET VARIABLES';
PROC PRINT DATA= VAR_INFO NOOBS;
   VAR DATA_SET VAR_NAME VAR_TYPE LABEL FORMAT;
RUN;

ODS PDF CLOSE;
```

As you can see, the temporary data set VAR_INFO has been created with information about the 130+ data sets in the SASHELP directory (ever been curious about all those SASHELP files?). The ODS statements refer to "PDF" instead of "HTML", but the coding is basically the same. This produces the output partially shown in Figure 4.

Figure 4



DATA_SET	VAR_NAME	VAR_TYPE	LABEL	FORMAT
ACCBW	language	char		
ACCBW	msgnum	num		
ACCBW	msgtext	char		
ACCBWMT	langu	char	Language field	
ACCBWMT	objvers	char	Object Version field	
ACCBWMT	tabname	char	BW Table	
ACCBWMT	where	char	Where Clause	
ACCPEO	msgnum	num		
ACCPEO	msgtext	char		
ACCPEO	msgtype	char		
ADOMSG	LEVEL	char		
ADOMSG	LINENO	num		
ADOMSG	MNEMONIC	char		
ADOMSG	MSGID	num		
ADOMSG	PBUTTONS	char		
ADOMSG	TEXT	char		
ADSMSG	LEVEL	char		
ADSMSG	LINENO	num		
ADSMSG	MNEMONIC	char		
ADSMSG	MSGID	num		
ADSMSG	PBUTTONS	char		
ADSMSG	TEXT	char		
AFMSG	LEVEL	char		
AFMSG	LINENO	num		
AFMSG	MNEMONIC	char		
AFMSG	MSGID	num		
AFMSG	PBUTTONS	char		

There are two levels to the directory structure in the left menu: "The Print Procedure" is shown because the name of the procedure used is the default value for ODS PROCLABEL; and "Data Set WORK.VAR_INFO" gives the name of the input data set. This output is fairly handy, with the right vertical scroll bar serving as a reasonably-convenient way to navigate among the rows which list the specified information for each data set in the SASHELP directory. But it is still a fairly forbidding list of information to scroll through, especially given the lack of specific information in the left menu.

We can easily make the left menu more informative, as well as enhance navigation among these files, by making a few simple modifications to the program code. In Figures 5 and 6 we see a number of improvements over Figure 4. First, as shown in the main window in Figure 4, there's a Table of Contents which lists each data set and its page number within the file. This is especially handy for hardcopies. Also, the directory in the left menu is now labeled "SASHELP Data Sets" instead of the default "The Print Procedure". The second level, instead of giving the name of the input data set, lists all the data sets. Although this repeats information in the Table of Contents, it allows for increased ease of navigation by (i) including a vertical scroll bar for this list, and (ii) each data set name here is a link to a data set, so you can click the data set you want and jump to a listing of the listed information, as shown in Figure 6.

It's interesting to compare the navigation within a PDF file to using hyperlinks to navigate among various HTML files. Often which output destination you choose will depend on other factors. For example, final documentation for a closed project might best be stored in PDF files if the documentation needs to be printed and delivered in hardcopy form.

The code that produces this output is shown immediately after Figure 6.

Figure 5

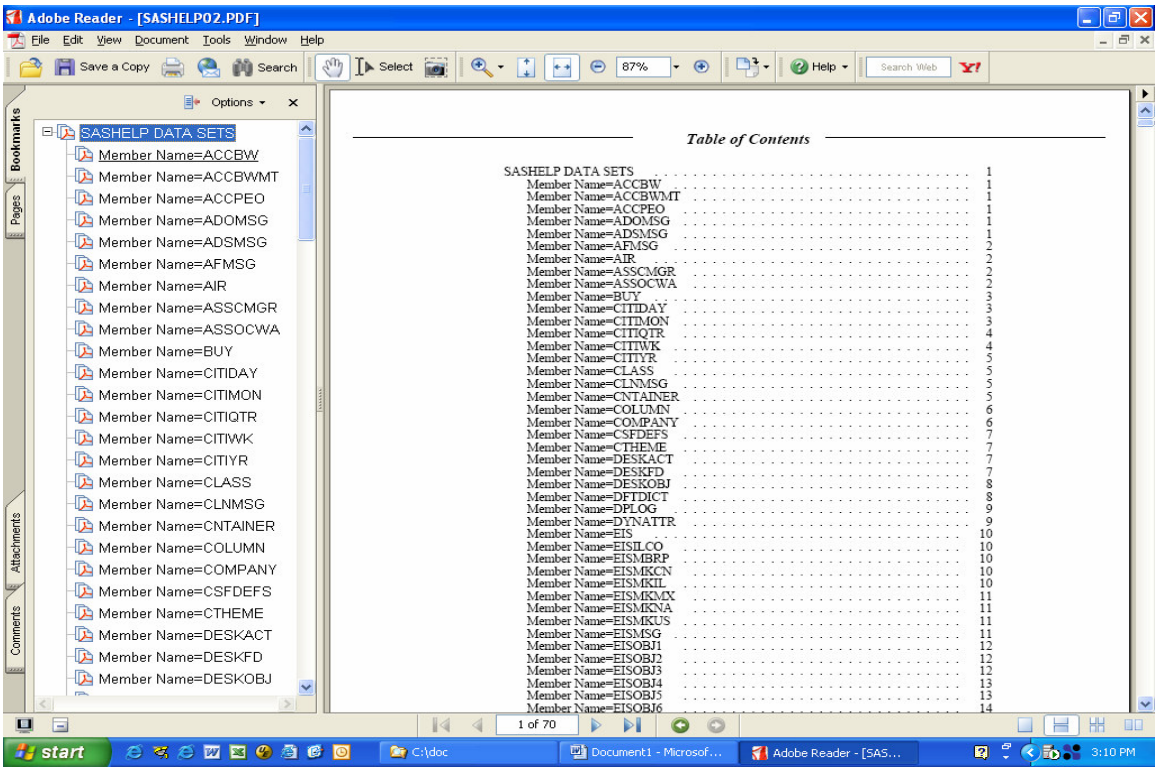
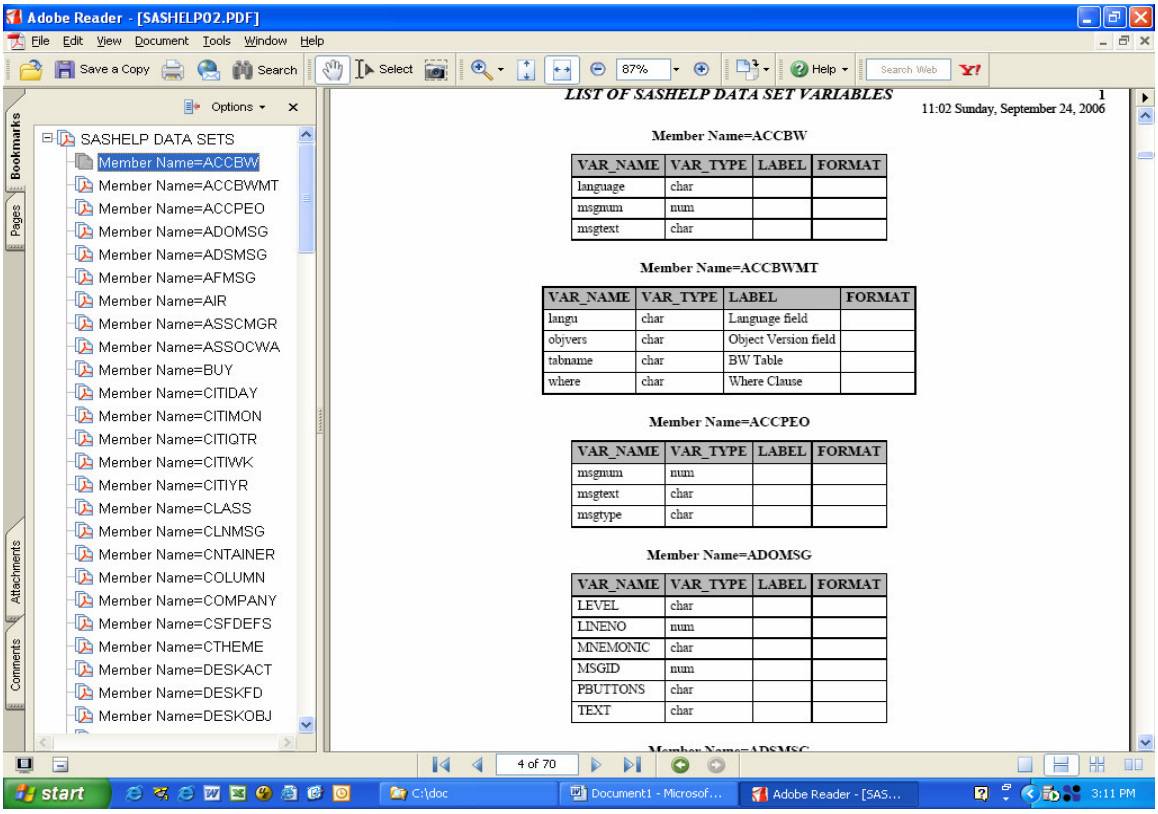


Figure 6



```
ODS PDF FILE='C:\DOC\SASHELP02.PDF' CONTENTS;
ODS PROCLABEL "SASHELP DATA SETS";

TITLE 'LIST OF SASHELP DATA SET VARIABLES';

PROC PRINT DATA= VAR_INFO NOOBS CONTENTS= '';
  VAR VAR_NAME VAR_TYPE LABEL FORMAT;
BY DATA_SET;
RUN;

ODS PDF CLOSE;
```

The first modification from the code that produced Figure 4 is the inclusion of "CONTENTS" in the ODS PDF statement. As you would expect, this causes SAS to create and display a Table of Contents as part of the output PDF file. We have also included here an ODS PROCLABEL statement which replaces the default name of the procedure used with the text "SASHELP DATA SETS".

There are two changes to the PROC PRINT specification. First we added the option CONTENTS=' '. This suppresses what would otherwise be an empty third level to the left menu stricture. Also, we have added a BY statement so that the output consists of one table for each data set, as shown in Figure 6. This is in contrast to generating one table with a row for each data set, as we saw in Figure 4.

As with HTML output, tailoring PDF output to meet your specific needs, although it requires learning some specific options such as CONTENTS for the ODS PDF statement, does not require putting aside your SAS knowledge and learning a new programming approach. Just as these are enhancements to SAS functionality, putting them to use requires only enhancing your knowledge of SAS.

Before moving on to discuss Excel output, we can illustrate the use of the PDF destination with output from statistical procedures (see Lund's SUGI 31 paper for a discussion of additional possibilities with PDF output). Figures 7 and 8 show output from a PROC REG example. The program code follows the figures.

Figure 7

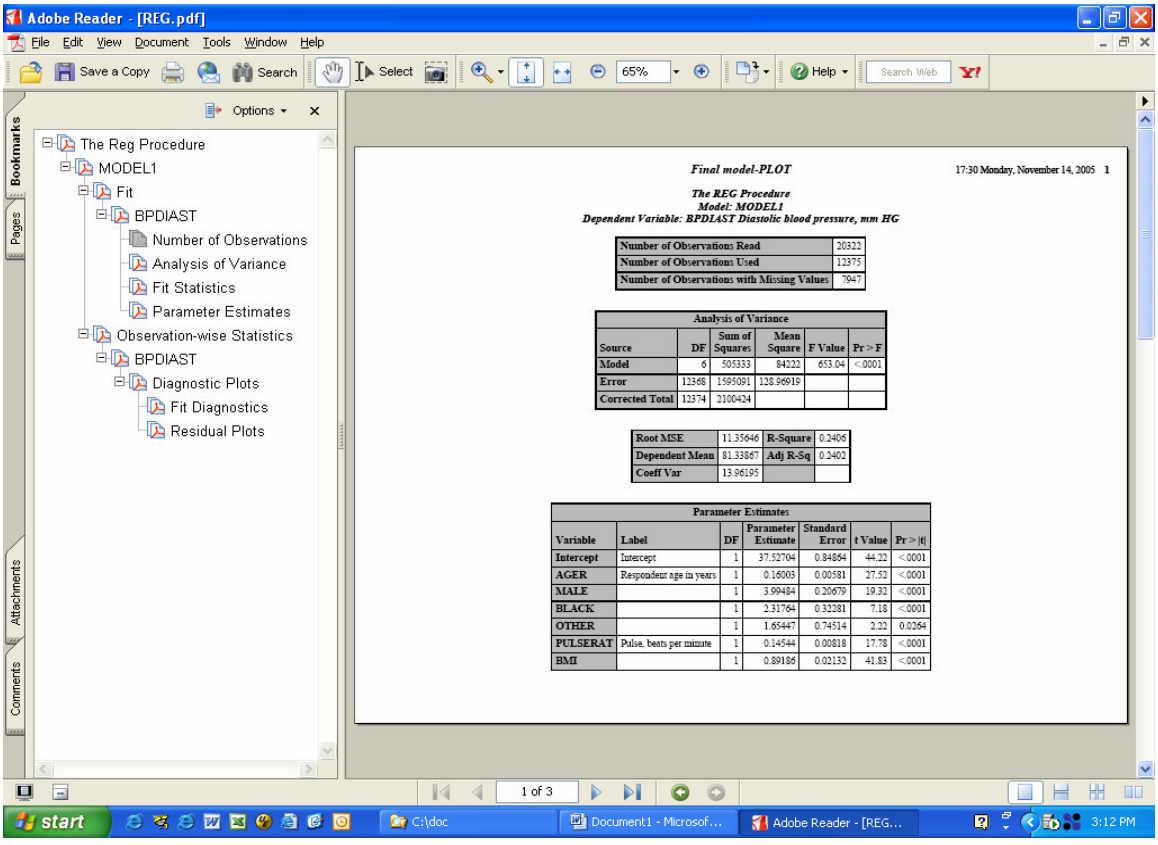
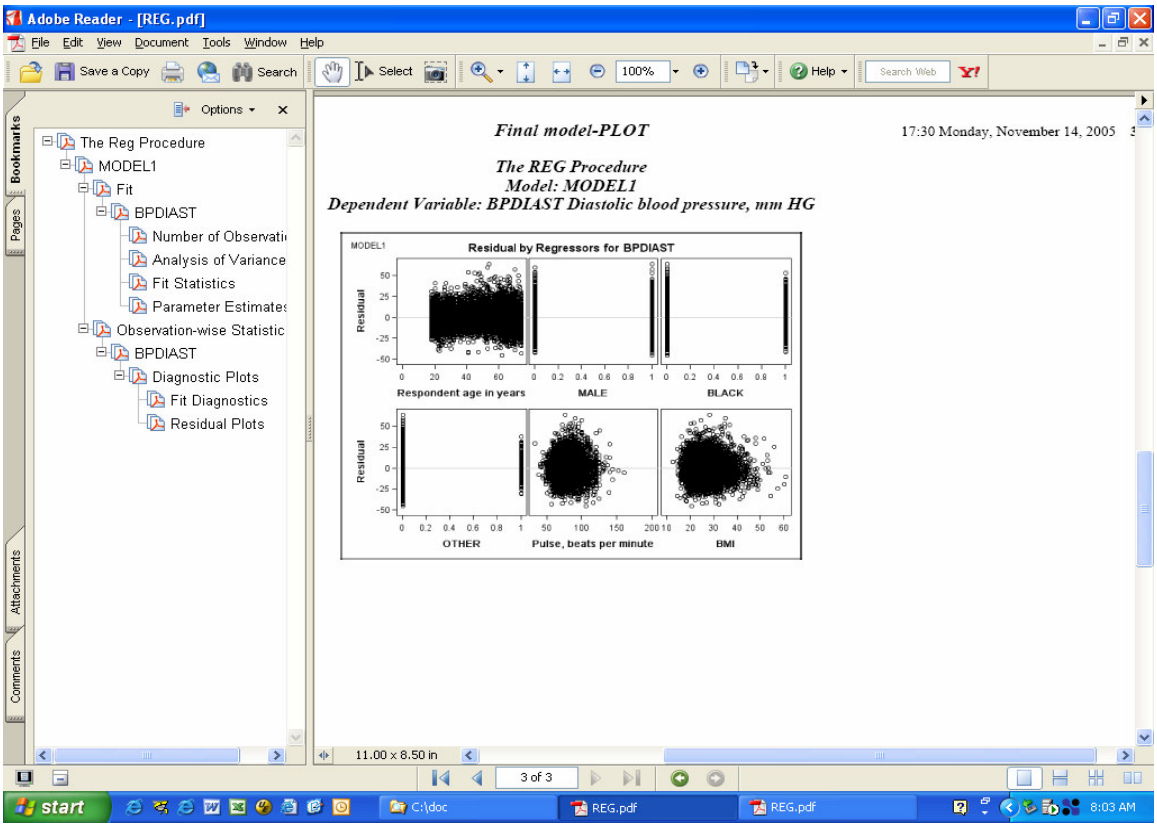


Figure 8




```
ODS PDF FILE='C:\DOC\REG.PDF';
ODS GRAPHICS ON;

TITLE 'FINAL MODEL-PLOT';

PROC REG DATA=BP2 PLOTS (MAXPOINTS=20000);
   MODEL BPDIAST=AGER MALE BLACK OTHER PULSERAT BMI;
RUN;

ODS GRAPHICS OFF;
ODS PDF CLOSE;
```

Aside from code that's specific to PROC REG, the only thing new here is the inclusion of the ODS GRAPHICS statements. A good introduction to ODS Graphics is given in the SUGI 31 paper by Rodriguez. An appendix lists all the statistical procedures which make use of this experimental ODS extension. As you can see in the left menu listing in Figures 7 and 8, the output consists of both tables and graphs. Having this type of output in PDF format is especially useful for documentation and cross-platform compatibility.

Although it is difficult to show in a static paper such as this, similar to the ease of navigation among files we see with HTML output, PDF output allows for easy navigation among tables within a single document. This is very useful for project data management and documentation as particular stages or modules of a project are completed, as well as when the project itself is finished.

In the next section we'll look at how a similar within-file ease of navigation can be achieved in Excel. The specifics will be different but the basic idea is the same: allow users to get to the data they want quickly and efficiently.

EXCEL OUTPUT

In my experience Excel is the most popular ODS destination among clients. In this section we will discuss using the ExcelXP tagset to generate Excel output. This tagset is well designed with easily understandable options you specify to tailor Excel properties to meet your needs. In general a tagset is simply a template that SAS uses to generate output from the ODS Markup destination. You may not know it but you have access to a large number of tagsets that come with SAS V9. To see which tagsets you have, just run the following code:

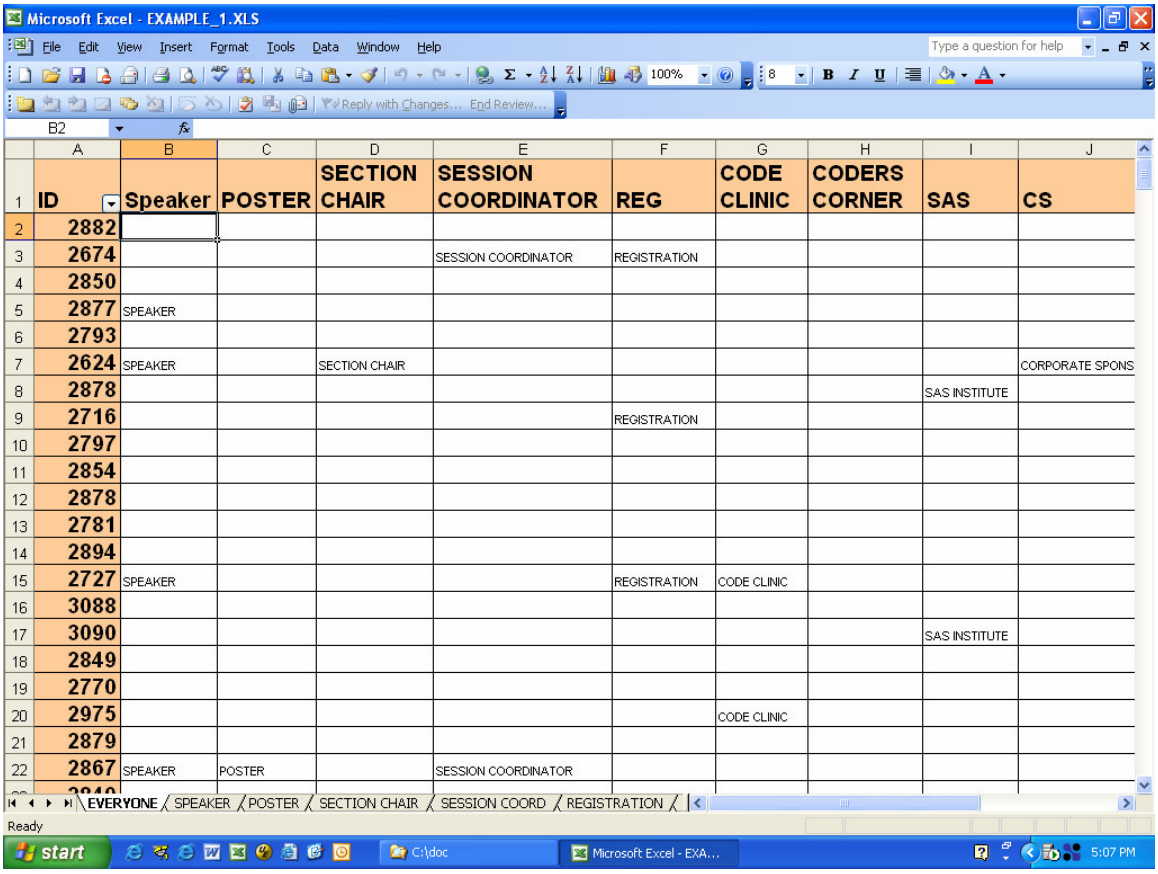
```
PROC TEMPLATE;
   LIST TAGSETS;
RUN;
```

The ExcelXP tagset is updated periodically and you can download the most-recent version at the SAS support site: http://support.sas.com/rnd/base/topics/odsmarkup/. Once you've downloaded the tagset files, you need to run the setup program EXCELXP-setup.sas. As with other ODS destinations, you will need to surround SAS code that generates output with the appropriate ODS statements.

```
ODS TAGSETS.EXCELXP
    PATH= 'C:\DOC\EXAMPLE_1.XLS'
    STYLE= XLSTATISTICAL
    OPTIONS ( ... ) ;

 ... SAS CODE THAT GENERATES OUTPUT ...

ODS TAGSETS.EXCELXP CLOSE;
```

The destination here, rather than HTML or PDF, is TAGSETS.EXCELXP. The path and filename are whatever you specify. The style here is XLSTATISTICAL but you can choose any style you have created or have access to. This is fairly basic ODS syntax so far. Let's take a look at the output in Figure 9.

Figure 9



	A	B	C	D	E	F	G	H	I	J
1	ID	Speaker	POSTER	SECTION CHAIR	SESSION COORDINATOR	REG	CODE CLINIC	CODERS CORNER	SAS	CS
2	2882									
3	2674				SESSION COORDINATOR	REGISTRATION				
4	2850									
5	2877	SPEAKER								
6	2793									
7	2624	SPEAKER		SECTION CHAIR						CORPORATE SPONS
8	2878								SAS INSTITUTE	
9	2716					REGISTRATION				
10	2797									
11	2854									
12	2878									
13	2781									
14	2894									
15	2727	SPEAKER				REGISTRATION	CODE CLINIC			
16	3088									
17	3090								SAS INSTITUTE	
18	2849									
19	2770									
20	2975						CODE CLINIC			
21	2879									
22	2867	SPEAKER	POSTER		SESSION COORDINATOR					

If you look at cell A1, you'll see that the Excel Autofilter option is being used. Also, what you can't see here, is that both the column and row headers are frozen, i.e. they remain visible as you scroll either horizontally or vertically.

```
ODS TAGSETS.EXCELXP
   PATH= 'C:\DOC\EXAMPLE_1.XLS'
   STYLE= XLSTATISTICAL
   OPTIONS (ORIENTATION='LANDSCAPE'
            AUTOFILTER='1'
            FROZEN_HEADER='YES'
            FROZEN_ROWHEADER='YES') ;

 ... SAS CODE THAT GENERATES OUTPUT ...
```

As you can see in the code above, the listed options allow you, within the SAS program, to specify properties of the output Excel worksheet. This greatly reduces post-processing in Excel. This is especially useful in contexts where a large number of spreadsheets are being generated. In the code above we've specified landscape orientation, as well as frozen column and row headers. In addition, the Excel Autofilter option is specified for the first column (A). This option can be used via a dropdown list to navigate among the rows of the file. This can be especially handy for analysts doing research or for use during conference calls.

If you look at the tabs across the bottom of Figure 9, you'll see that there are a number of different sheets. These were all generated by the same program. If you look again at the code above you'll see that there isn't an ODS CLOSE statement. Until this destination is closed, the specifications and options shown are still in effect. To generate the first sheet (EVERYONE) we would add the following:

```
ODS TAGSETS.EXCELXP
   OPTIONS (SHEET_NAME='EVERYONE');

PROC PRINT DATA= EX1;
ID ID;
VAR SPEAKER POSTER ... ;
RUN;
```

The effect of this code, following that above, is to add the specification for sheet name to the options already specified. The Excel output from the PROC PRINT will make use of these specifications. Note that the ID statement is used to specify the column for the frozen row headers.

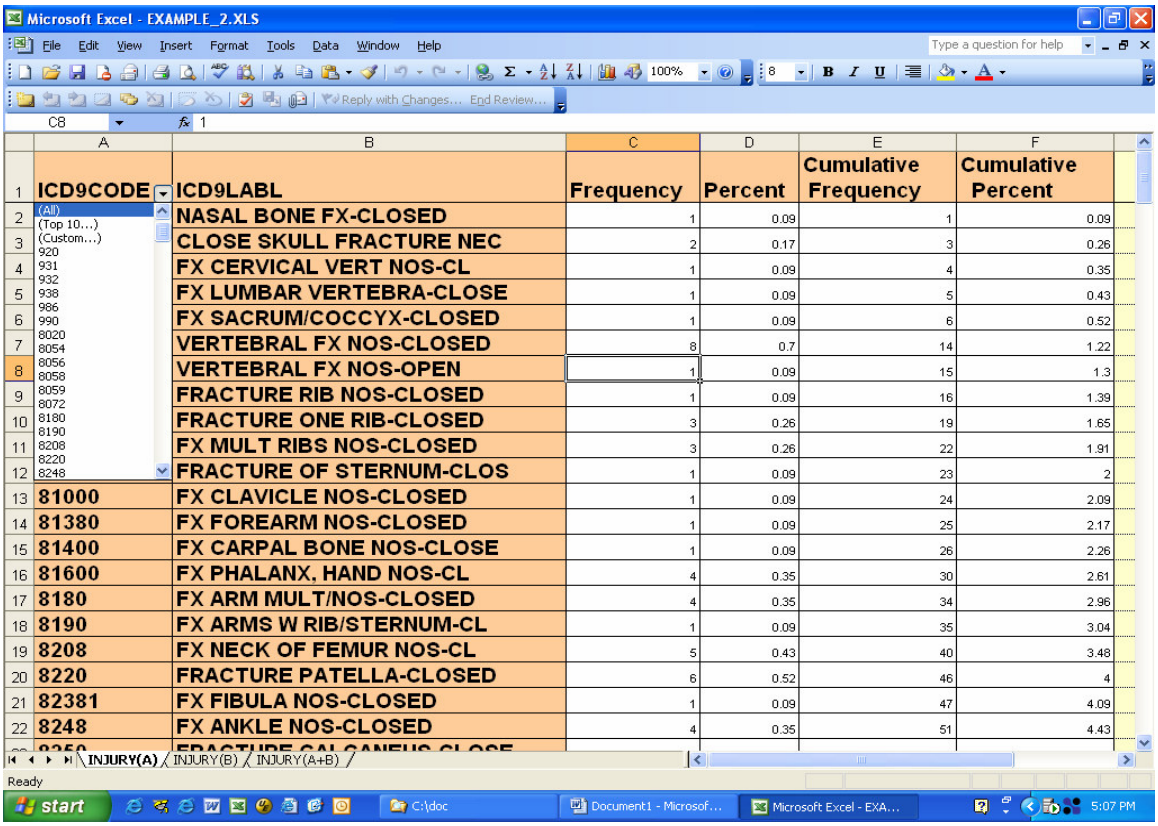
We can continue to generate worksheets within the EXAMPLE_1.XLS file by altering options where needed. Since a new PATH is not specified, all output will still be to this file, but to additional worksheets.

```
ODS TAGSETS.EXCELXP
   OPTIONS (SHEET_NAME='SPEAKER');

PROC PRINT DATA= EX1;
   VAR ID;
   WHERE SPEAKER IS NOT MISSING ;
RUN;

ODS TAGSETS.EXCELXP
   OPTIONS (SHEET_NAME='POSTER');

PROC PRINT DATA= EX1;
   VAR ID;
   WHERE POSTER IS NOT MISSING ;
RUN;
```

This code generates two different worksheets, one named SPEAKER and one named POSTER. In this example the sheets only contain the ID values of those records satisfying the condition of the WHERE clause, but it should be clear how easy this makes it to output an Excel file that (i) makes use of the Autofilter option to make navigation among rows quick and easy, but also (ii) includes conveniently grouped subsets of data or metadata in an array of worksheets. Note that this is but one way to generate multi-sheet workbooks (see DelGobbo's SUGI 31 paper).

Let's take a look now at a similar example, one that lists frequencies for various types of injuries by ICD-9 codes and corresponding labels. The Autofilter dropdown menu is partially shown in Figure 10. This file has three worksheets: INJURY(A), INJURY(B) and INJURY(A+B).

Figure 10



Microsoft Excel - EXAMPLE_2.XLS

	A	B	C	D	E	F
1	ICD9CODE	ICD9LABL	Frequency	Percent	Cumulative Frequency	Cumulative Percent
2	(All)	NASAL BONE FX-CLOSED	1	0.09	1	0.09
3	(Top 10...)	CLOSE SKULL FRACTURE NEC	2	0.17	3	0.26
4	(Custom...)	FX CERVICAL VERT NOS-CL	1	0.09	4	0.35
5	920	FX LUMBAR VERTEBRA-CLOSE	1	0.09	5	0.43
6	931	FX SACRUM/COCCYX-CLOSED	1	0.09	6	0.52
7	932	VERTEBRAL FX NOS-CLOSED	8	0.7	14	1.22
8	938	VERTEBRAL FX NOS-OPEN	1	0.09	15	1.3
9	986	FRACTURE RIB NOS-CLOSED	1	0.09	16	1.39
10	990	FRACTURE ONE RIB-CLOSED	3	0.26	19	1.65
11	8020	FX MULT RIBS NOS-CLOSED	3	0.26	22	1.91
12	8054	FRACTURE OF STERNUM-CLOS	1	0.09	23	2
13	81000	FX CLAVICLE NOS-CLOSED	1	0.09	24	2.09
14	81380	FX FOREARM NOS-CLOSED	1	0.09	25	2.17
15	81400	FX CARPAL BONE NOS-CLOSE	1	0.09	26	2.26
16	81600	FX PHALANX, HAND NOS-CL	4	0.35	30	2.61
17	8180	FX ARM MULT/NOS-CLOSED	4	0.35	34	2.96
18	8190	FX ARMS W RIB/STERNUM-CL	1	0.09	35	3.04
19	8208	FX NECK OF FEMUR NOS-CL	5	0.43	40	3.48
20	8220	FRACTURE PATELLA-CLOSED	6	0.52	46	4
21	82381	FX FIBULA NOS-CLOSED	1	0.09	47	4.09
22	8248	FX ANKLE NOS-CLOSED	4	0.35	51	4.43

Given the code shown above that was used to generate Figure 9, it should be clear how these sheets were generated. Instead of PROC PRINT a 2-way frequency table (ICD9CODE*ICD9LABL) was generated by PROC FREQ. As with Figure 9, WHERE clauses were used to subset to the required rows for output. Although Excel isn't often chosen for its navigational advantages, we can see that it's an excellent choice for users who are used to Excel, and who need to have data available in a familiar and handy format. In the two examples illustrated here, specific data files were used, but just as we saw with HTML and PDF output, project-level data and metadata would also be appropriate for inclusion in Excel worksheets.

CONCLUSION

This paper has used concrete examples to illustrate the advantages of the expanded set of output destinations that ODS makes available. Any data management or documentation system needs to be designed with its intended users in mind. In the real workplace everyone is busy and impatient with systems that aren't quick and intuitive. We've discussed three choices made available by SAS' Output Delivery System: HTML, PDF and Excel. Each has its own advantages with respect to data management and documentation.

But more important than the specific examples illustrated here (which only show a small subset of the possibilities) is the basic approach of taking stock of the choices that SAS now has to offer with respect to output destinations, and then designing sets of connected files, and tables within files, that allow for quick access to important project data.

The basic approach is to create an initial 'entry' file that, once accessed, contains connections to the relevant project data. These connections may take the form of HTML hyperlinks which allow for easy navigation among separate HTML files. They may also be links within a PDF document that allow quick jumps to specific tables. Within Excel the Autofilter option allows for dropdown menus for record-specific navigation. Also, the ability within SAS to generate multi-sheet workbooks provides conveniently organized and labeled project information within one file.

The advantages of this approach from a usability standpoint are clear. But the other important advantage is that creating this type of data management and documentation system builds on the skills you and your colleagues already possess as SAS programmers. That's your classic win-win situation.

REFERENCES

DelGobbo, Vince. Creating *AND* Importing Multi-Sheet Excel Workbooks the Easy Way with SAS. SUGI 31 (Paper 115-31).

Davis, Michael. You Could Look It Up: An Introduction to SASHELP Dictionary Views. SUGI 26 (Paper 17-26).

DiIorio, Frank & Abolafia, Jeff. Dictionary Tables and Views: Essential Tools for Serious Applications. SUGI 29 (Paper 237-29).

Haworth, Lauren. SAS with Style: Creating your own ODS Style Template for PDF Output. SUGI 30 (Paper 132-30).

Haworth, Lauren. *Output Delivery System: The Basics*. SAS Publishing, Books by Users Press.

Lund, Pete. PDF Can be Pretty Darn Fancy: Tips and Tricks for the ODS PDF Destination. SUGI 31 (Paper 092-31).

Rodriquez, Robert N. An Introduction to ODS for Statistical Graphics in SAS 9.1. SUGI 31 (paper 204-29).

The ExcelXP Tagset and Microsoft Excel. Demo and sample code available at http://support.sas.com/rnd/base/topics/odsmarkup/excelxp_demo.html.

Usability.gov. A Web site maintained by the Federal government as a resource for information on usability and user-centered design.

ACKNOWLEDGMENTS

I have benefited from the SAS expertise of my colleagues at Social & Scientific Systems Inc., as well as from DCSUG, NESUG and SUGI presentations on ODS, data management and PROC SQL. Some of the material included here was presented at a DCSUG meeting and I'm grateful to the audience for helpful comments. Thanks to Xiuhua Chen for the PROC REG / PDF example.

CONTACT INFORMATION

I would appreciate any comments or questions. Please contact me at:

Social & Scientific Systems, Inc.
8757 Georgia Avenue, 12th Floor
Silver Spring, MD 20910
E-mail: pgorrell@s-3.com