ODS, YES! Odious, NO! – An Introduction to the SAS Output Delivery System

Lara Bryant, University of North Carolina at Chapel Hill, Chapel Hill, NC
Sally Muller, University of North Carolina at Chapel Hill, Chapel Hill, NC
Ray Pass, Ray Pass Consulting, Hartsdale, NY

ABSTRACT
ODS (originally pronounced 'odious', but now pronounced 'ya gotta love it') is the new SAS System facility, starting with Version 7, that you can use to format your PROC and DATA output in ways just recently only dreamed about. ODS offers greatly enhanced flexibility and ease of use for both new SAS users and experienced SAS users new to ODS. This paper will discuss the basics of ODS, emphasizing methods of converting standard PROC output to the following "destinations":

- **Listing**	- default (the only way to get PROC output up to now)
- **HTML**	- HyperText Markup Language (probably the best tool available for information exchange today)
- **Output**	- SAS data sets (no more PROC PRINTTO!)
- **Printer**	- available experimentally in V7, and for production in V8. Produces both Postscript and PCL output on all hosts, and on PC hosts additionally produces output for any printer supported by the host operating system. Note with Version 8.1, PDF (Postscript Display Format) is also available as production.
- **RTF**	- for importing into MS Word. (in production now with V8.1, but not covered in this paper.

For more information on RTF see:

http://www.sas.com/rnd/base/news/odsrtf/index.html

Also not covered in this paper, the following destinations are available as experimental in Version 7 and Version 8:

- **LaTex**	- a driver that produces your output marked up using LaTex.
- **XML**	- a driver that produces XML
- **HTML STYLESHEET**	- lets you use HTML CSS (Cascading Style Sheets)

For more information on these experimental destinations see:

http://www.sas.com/rnd/base/topics/expv8/index.html

Prior to ODS, all SAS output results were lumped together in a single "listing" output. With the advent of ODS, each PROC now produces one or more data components which are then combined with different formats to produce one or more output objects. These output objects are then sent to one or more destinations as defined above. In this paper we will demonstrate how you select the output objects to send to each destination, and the syntax for each destination. By the end of the paper you will have a working knowledge of ODS and feel comfortable enough to easily create at least three new kinds of output in SAS!

INTRODUCTION
Creating output objects that can be sent to destinations (e.g. HTML) is often just a matter of running procedures in your existing SAS program with just a few extra lines of code (sometimes only one line). When you run a procedure or DATA step, ODS combines the resulting data with a template (or table definition) to create an output object, or a series of output objects coming from various parts of the procedure's output. ODS allows you to choose specific output objects created from a procedure or DATA step to send to an output destination. ODS provides default table definitions for most (but not all!) procedures and for the DATA step. You can also create or modify your own table definition with PROC TEMPLATE. The output object is formatted according to its content and the destination you send it to. You can also send your output to more than one destination. For example, you can create an output data set from a PROC MEANS procedure that is also displayed on an HTML page.

OPENING AND CLOSING DESTINATIONS
The Listing, HTML, and Output destinations can be open or closed. By default, the Listing destination is open, and the HTML, Output, and Printer destinations are closed. The statement for opening the Listing destination is:

```
ods listing;
```

The commands for opening the HTML, Output, and Printer destinations are more detailed, and therefore are presented in this paper in the overview of each destination. To close a destination, the syntax is

```
      ods <destination> close;
e.g.  ods listing close;
```

You may want to close the Listing destination to free up resources that would otherwise be used to send output objects to this destination.

SELECTION AND EXCLUSION LISTS
For each destination, the SAS System maintains a list of the objects that are to be sent there. The SAS System also maintains an overall list of objects that are to be sent to all open destinations. If you are selecting objects to send to a destination, SAS maintains a SELECTION list. If you are selecting objects that you do not want sent to a destination, SAS maintains an EXCLUSION list for that destination. Generally you need only select or exclude objects for a particular destination, rather than trying to maintain both a SELECTION and an EXCLUSION list for that destination. The same holds true if you are creating an overall selection or exclusion list -- you only need one or the other.

There are two ways that these SELECTION and EXCLUSION lists can be modified:

- explicit modification from a command by you
- automatic modification by ODS at certain points (step boundaries) in the SAS program

For more information on step boundaries see "SAS Language Reference Concepts Version 8," pg. 271.

Explicit Modification
To explicitly modify the overall SELECTION and EXCLUSION lists, you may use the following syntax:

```
ods <options>;
```

To explicitly modify a specific destination's SELECTION and EXCLUSION lists, you may use the following syntax:

```
ods listing <options>;
ods html    <options>;
ods printer <options>;
```

where the options are

```
select  <specific output objects>
select  all
select  none
exclude <specific output objects>
exclude all
exclude none
```

The default values for the destinations are as follows:

Overall list	- `select all`
Listing destination	- `select all`
HTML destination	- `select all`
Printer destination	- `select all`
Output destination	- `exclude all`

Changing the overall list is helpful if you want to exclude an object from all destinations. For example, rather than typing,

```
ods html exclude all;
ods printer exclude all;
```

You could simply type:

```
ods exclude all;
```

Automatic Modification

When you do NOT explicitly modify a SELECTION or EXCLUSION list, ODS automatically sets defaults as noted above at every step boundary. (A step boundary signals the end of the preceding step, for instance a "run;" statement or a "quit;" statement or a new DATA or PROC step.) When you do use explicit modification, ODS, by default, maintains the modifications for *one use only*, reverting back to defaults at the boundary. This can be overcome by using the PERSIST option.

Persist Option with SELECT OR EXCLUDE

Consider the following code:

```
ods listing select BasicMeasures;

proc univariate data='A:/meddat';
run;
proc univariate data='A;/meddat';
run;
```

As a result of this code, ODS would select only the "BasicMeasures" statistics for the first PROC UNIVARIATE. The RUN statement ends the procedure (this is a step boundary, but even if you did not specify the RUN statement, the beginning of the second PROC UNIVARIATE would end the first PROC UNIVARIATE). Either way, you would only have "BasicMeasures" printed for the first PROC. After the first PROC, the list is automatically set to its default value, which is SELECT ALL for the default Listing destination. The second PROC would therefore include all statistics generated by the PROC UNIVARIATE. Obviously, if you only wanted "BasicMeasures" throughout, it would be tedious to have to specify the desired list after every procedure. ODS provides a way around this. By adding the PERSIST option to the SELECT/ EXCLUDE statement, you only have to specify the SELECTION/ EXCLUSION list once for it to be maintained throughout the program (or at least until the next encountered SELECT or EXCLUDE command). So if we run the following code

```
ods listing select BasicMeasures (persist);
proc univariate data='A:/meddat';
run;
proc univariate data='A;/meddat';
run;
```

the BasicMeasures statistics will be selected for both PROC UNIVARIATEs. The PERSIST option can also be used for an HTML or Printer list, for example:

```
ods html select BasicMeasures (persist);
```

The PERSIST syntax for the Output destination is more involved, and is explained in the Output destination section of this paper.

Resetting Lists for RUN-Group Processing

In the previous examples, a RUN statement would end the PROC and reset the SELECTION/EXCLUSION list to the default value if the PERSIST option was not specified. However, there are several procedures that are not terminated by a RUN statement (RUN-Group processing), such as PROC DATASETS, PROC GLM, and PROC REG. In these cases, unless a QUIT statement is encountered, the PROC will continue to run. This may produce some unexpected results on your SELECTION/EXCLUSION list. For example, consider the following code, (the FILE= option is discussed below in "file types"):

```
ods html file='A:/new.htm';
ods html select Anova;

proc reg data='A:/aggrec';
   model inpdol=age;
run;

ods html select FitStatistics;

proc reg data='A:/aggrec';
   model outpdol=age;
run;

ods html close;
```

In the program above, ODS would create "Anova" statistics for the first PROC REG. This would remain intact through the RUN statement because a RUN statement does not end a running PROC REG. When ODS reaches the second PROC REG, it would end the first PROC and set the SELECTION list to its default value of SELECT ALL. Therefore, rather than having the desired "FitStatistics" for the last PROC REG, ODS would create ALL the statistics. The simple solution is to specifically end the first PROC REG with a QUIT statement as follows (the SHOW statement is discussed below):

```
ods html file='A:/new.htm';
ods html select Anova;

proc reg data='A:/aggrec';
   model inpdol=age;
run;
ods html show;
quit;

ods html show;
ods html select FitStatistics;

proc reg data='A:/aggrec';
   model outpdol=age;
run;

ods html show;
quit;

ods html close;
```

This program produces the desired results: "Anova" statistics for the first PROC REG and "FitStatistics" for the second PROC REG.

ODS SHOW STATEMENT

At any point in your program, you can use the ODS SHOW to see what ODS has on the SELECTION/EXCLUSION list for a specific destination. The following syntax,

```
ods <destination> show;
```

requests that the SELECTION/EXCLUSION list for a particular destination appear in the log. If no destination is specified, the OVERALL list is displayed. In the example immediately above, the log would contain

```
 Current HTML select list is:
 1. Anova
```

after the first SHOW statement, and

```
 Current HTML select list is:
 1. FitStatistics
```

after the last one.

ODS TRACE STATEMENT

Part of the power of ODS is that you can indicate which output objects to create, and even tell ODS to send the output objects created by the same procedure to different destinations. For example, rather than all of the statistics, you may want only the mean, standard deviation, and median generated by the PROC UNIVARIATE. However, in order to specify which output objects to select, you must know the name of the object produced by your SAS program. ODS provides a method of viewing the name of each output object created. The syntax,

```
 ods trace on < / listing | label > ;
```

displays a "trace record" of the name and other information about each output object produced by the program in the SAS log. The LISTING option instructs ODS to put the trace record directly above the output to which it refers. This is extremely useful for determining the name and path of the output objects of interest. The LABEL option instructs ODS to include the label path in the trace record. The code below illustrates both the TRACE statement and the LABEL option:

```
ods trace on / label;
proc univariate;
    var meddol;
run:
ods trace off;
```

The SAS Log that results from this program is shown below. Note that you not only get the name of the output object, but since you specified the label option, you also get the label path of the output object:

```
Output Added:
-------------
Name:      Moments
Label:     Moments
Template:  base.univariate.Moments
Path:      Univariate.meddol.Moments
Label Path:"The Univariate Procedure"."meddol".
           "Moments"
-------------

Output Added:
-------------
Name:      BasicMeasures
Label:     Basic Measures of Location and
           Variability
Template:  base.univariate.Measures
Path:      Univariate.meddol.BasicMeasures
Label Path:"The Univariate Procedure"."meddol".
           "Basic Measures of Location and
           Variability"
-------------

Output Added:
-------------
Name:      TestsForLocation
Label:     Tests For Location
Template:  base.univariate.Location
Path:      Univariate.meddol.TestsForLocation
Label Path:"The Univariate Procedure"."meddol".
           "Tests For Location"
-------------
```

```
Output Added:
-------------
Name:      Quantiles
Label:     Quantiles
Template:  base.univariate.Quantiles
Path:      Univariate.meddol.Quantiles
Label Path:"The Univariate Procedure"."meddol".
           "Quantiles"
-------------

Output Added:
-------------
Name:      ExtremeObs
Label:     Extreme Observations
Template:  base.univariate.ExtObs
Path:      Univariate.meddol.ExtremeObs
Label Path:"The Univariate Procedure"."meddol".
           "Extreme Observations"
-------------
```

Fortunately, although you can then specify the output object by using the full path name, you can also specify the output object by using any part of the path that begins immediately after a period and continuing to the end. For example, if you want to send the Quantiles and Moments for all variables to a web page, you could enter,

```
 ods html select quantiles moments;
```

or if you just want the Quantiles and Moments for the MEDDOL variable only:

```
 ods html select meddol.quantiles
                 meddol.moments;
```

The label path can be used in the same way. You can also specify an output object with a mixture of labels and paths, such as

```
 ods html select meddol."quantiles";
```

Often it is easier to select the variables in the PROC step, and the desired statistics in the ODS step. For example, rather than typing,

```
 ods html select meddol.quantiles
               inpdol.quantiles
               hosp.quantiles
               ambul.quantiles;
 proc univariate;
 run;
```

an easier method that gives the same results would be

```
 ods html select quantiles;
 proc univariate;
    var meddol inpdol hosp ambul;
 run;
```

Note, once you "turn on" the ODS TRACE ON statement in your SAS session, ODS will continue to write trace records until you issue the statement ODS TRACE OFF. This means that if you start a new procedure or even a new program in the same SAS session, TRACE ON will be in effect, until you turn it off. Also note that you must have a run statement between the TRACE ON and TRACE OFF statements in order for the trace record to be created.

ODS HTML DESTINATION

File types

The HTML destination can produce four kinds of files (web pages):

1) **BODY** file: This is a required file that contains the output object(s) generated from the PROCs or DATA steps. Basically, this is where you store the results that will ultimately be displayed on your HTML report or

web site. If your SAS job creates an output object that is routed to an HTML destination, ODS places the results within HTML <TABLE> tags, where they are stored as one or more HTML tables. If your SAS job creates a graphic object, the BODY file has an <IMG> (image) tag that references graphic output objects. Note that the BODY file can be specified with either the BODY= or the FILE= parameter.

2) **CONTENTS** file: The CONTENTS file contains a link to each of the output objects that are stored in the BODY file, and is specified by the CONTENTS= parameter.

3) **PAGE** file: This is useful if you have a lot of output, and you do not want it to all be stored on one long page. The PAGE file contains a link to each separate page (of the BODY file) of HTML output that ODS creates from a PROC or DATA step. The PAGE file is similar to the CONTENTS file, except that the CONTENTS file has a link to each output object, whereas the PAGE file has a link to each page of output that is created. The CONTENTS and PAGE files will be identical if you specify in the NEWFILE parameter that you would like each output object placed on a separate BODY file. An example that illustrates the NEWFILE parameter is presented later in this paper. You specify the PAGE file with the PAGE= parameter.

4) **FRAME** file: Provides a simultaneous view of all files included in the ODS HTML statement. You specify the FRAME file with the FRAME= parameter.

The syntax for creating these files is

```
ods html file-type = 'file-specification'
                        <(parameters)>;
```

Here is an example of ODS HTML statements which generate a BODY file and a CONTENTS file:

```
ods html body     = 'c:\temp\body.htm'
         contents = 'c:\temp\contents.htm';
```

In the above code, the BODY file could also have been specified with a FILE= parameter. Note that the BODY file, and only the BODY file, is *required* as an HTML output destination.

Additional HTML Parmeters

1**PATH=** : As mentioned, you use the BODY= parameter to tell ODS where to store an HTML file. In addition, you can use PATH= to tell ODS in what directory to store all the HTML files that you create. The PATH= option may refer to an external (quoted) file specification, a SAS fileref or a SAS libname.catalog. For example,

```
ods html path = 'C:\MyDocuments'
         body = 'body.htm'
         contents = 'contents.htm';
```

Note that if you use the PATH= statement, you must do so before specifying the HTML pages.

2) **URL=** sub-parameter: You can improve on PATH= by including a Uniform-Resource-Locator (URL) sub-parameter that will use the given URL instead of the file name for all the links and references that it creates to the file. This is helpful if you want to create a FRAME file, and/or will be moving the files around. For example:

```
ods html path = 'C:\MyDocuments'
         (url = 'http://www.unc.edu/~jismith')
         body = 'body.htm'
         contents = 'contents.htm';
```

Note that the URL= sub-parameter of the PATH= option is enclosed in parentheses.

You can also specify the URL= sub-parameter in the parameter for the BODY file, as in the following:

```
ods html path ='C:\MyDocuments '
         body ='body.htm'
         (url ='http://www.unc.edu/~jismith');
```

The results will be identical.

3) **ANCHOR=** : Each output object in the BODY file is identified by an HTML <ANCHOR> tag. These anchor tags allow the CONTENTS, PAGE and FRAME files to link to, or reference the output objects in the BODY file. You can change the base name for the HTML anchor tags with the ANCHOR= parameter. The syntax for this option is:

```
anchor = 'anchor-name';
```

Since each anchor name in a file must be unique, ODS will automatically "increment" the name that you specify. For example, if you specify

```
anchor = 'tabulate';
```

ODS names the first anchor TABULATE. The second anchor is named TABULATE1; the third is named TABULATE2, and so on. The anchor names are only of interest to you if you need to write to the HTML page; otherwise you need not concern yourself with them. However, you do need to remember to *always* specify a new anchor name each time you open the BODY file so that the same anchor tags are not written to the file again.

4) **NO_TOP_MATTER** and **NO_BOTTOM_MATTER** parameters: These parameters circumvent the default action of writing some HTML to the top and bottom of the file that is open for HTML output. The benefit of these parameters is that the HTML BODY page is "cleaner" when viewed by the browser.

5) Descriptive text parameter: This parameter allows you to include comments in between the output of your PROCs. You specify the descriptive text inside parentheses next to the BODY=, CONTENTS=, PAGE=, or FRAME= options. Adding comments to your HTML page is helpful for many reasons. For example, you might like to point out some of the interesting results you obtained.

EXAMPLE 1, Putting it all together. The following code places output from several procedures on the same HTML page and uses many of the HTML parameters discussed above - including incorporating descriptive text between the output objects. The SAS statements are numbered for comments following the code:

```
1)  libname  health 'C:Data';
2)  filename web    'C:\Data\body.htm';
3)  ods listing close;
4)  ods html path = 'C:\Data'
         (url = 'http://www.unc.edu/~jismith/')
         body = web (no_bottom_matter);
5)  proc univariate data=health.meddat;
       var inpdol outpdol;
6)  run;
7)  ods html close;
8)  filename web 'C:\Data\body.htm' mod;
9)  data _null_;
10)    file web;
11)    put '<h3> We want to put comments in
            after the first procedure. </h3>';
12) run;
13) ods html body   = web (no_top_matter
                           no_bottom_matter)
             anchor = 'univ';
14) proc freq data=health.meddat;
       table site;
15) run;
16) ods html close;
17) data _null_;
       file Web;
```

```
      put '<h3> We also want comments after
           the second procedure is run.</h3>';
18) run;
19) ods html body   = web (no_top_matter)
           anchor = 'freq';
20) ods html close;
```

And now the comments:

(1) Identifies the location of the SAS catalog (C\:Data) containing the SAS data set used for the PROCS.

(2) The FILENAME statement creates a fileref (WEB) for the BODY file, where all the output will be stored. Recall the default list for HTML is SELECT ALL. Since no selection commands are specified, everything included in the program will be sent to the HTML file at 'C:\Data\body.htm'

(3) The Listing destination is closed to free up resources.

(4) The NO_BOTTOM_MATTER option suppresses any default HTML at the bottom of ' C:/Data/body.htm'

(5) All statistics created by PROC UNIVARIATE will be generated for the variables INPDOL and OUTPDOL.

(6) Remember that a RUN statement goes after the PROC, and before closing the HTML destination.

(7) The HTML destination must be closed to append to it later.

(8) This references the BODY file used above, and MOD indicates that we want to append to the file.

(9) This DATA _NULL_ step writes some descriptive HTML code to the BODY file via the PUT and FILE statements.

(13) This opens the HTML destination 'C:\Data\body.htm' as identified by the fileref WEB, and suppresses any default HTML code on the top and bottom of the file. The ANCHOR= option creates a base name for the HTML anchor tags. You should *always* specify a new anchor name each time you open the BODY location so that the same anchor tags are not written to the file again.

(19) Open the HTML destination again in order for the new output to be written to the HTML file. The ANCHOR statement provides a new base name.

The resulting HTML page is shown below. The name of the HTML file that is created is 'body.htm' and it is stored in 'C:\Data'. Since we did not specify a template, the default template is used.

The UNIVARIATE Procedure
Variable: INPDOL

Moments			
N	1000	**Sum Weights**	1000
Mean	247.50903	**Sum Observations**	247509.03
Std Deviation	1354.43565	**Variance**	1834495.93
Skewness	9.68499218	**Kurtosis**	119.342445
Uncorrected SS	1893922159	**Corrected SS**	1832661439
Coeff Variation	547.226762	**Std Error Mean**	42.831016

Basic Statistical Measures			
Location		**Variability**	
Mean	247.5090	**Std Deviation**	1354
Median	0.0000	**Variance**	1834496
Mode	0.0000	**Range**	23018
		Interquartile Range	0

Tests for Location: Mu0=0				
Test	**Statistic**		**p Value**	
Student's t	T	5.778734	Pr > |t|	<.0001
Sign	M	43.5	Pr >= |M|	<.0001
Signed Rank	S	1914	Pr >= |S|	<.0001

Quantiles (Definition 5)	
Quantile	**Estimate**
100% Max	23018.39
99%	5618.30

95%	1538.71
90%	0.00
75% Q3	0.00
50% Median	0.00
25% Q1	0.00
10%	0.00
5%	0.00
1%	0.00
0% Min	0.00

Extreme Observations			
Lowest		**Highest**	
Value	**Obs**	**Value**	**Obs**
0	1000	11367.8	722
0	999	12175.4	983
0	998	13119.0	162

The UNIVARIATE Procedure
Variable: OUTPDOL

Moments			
N	1000	**Sum Weights**	1000
Mean	111.39919	**Sum Observations**	111399.19
Std Deviation	248.217815	**Variance**	61612.0838
Skewness	9.53237587	**Kurtosis**	139.188553
Uncorrected SS	73960251.2	**Corrected SS**	61550471.7
Coeff Variation	222.81833	**Std Error Mean**	7.84933652

Basic Statistical Measures			
Location		**Variability**	
Mean	111.3992	**Std Deviation**	248.21782
Median	44.0000	**Variance**	61612
Mode	0.0000	**Range**	4523
		Interquartile Range	107.15000

Tests for Location: Mu0=0				
Test	**Statistic**		**p Value**	
Student's t	t	14.19218	Pr > |t|	<.0001
Sign	M	390.5	Pr >= |M|	<.0001
Signed Rank	S	152685.5	Pr >= |S|	<.0001

Quantiles (Definition 5)	
Quantile	**Estimate**
100% Max	4522.680
99%	970.105

95%	412.975
90%	267.525
75% Q3	117.150
50% Median	44.000
25% Q1	10.000
10%	0.000
5%	0.000
1%	0.000
0% Min	0.000

Extreme Observations

Lowest		Highest	
Value	**Obs**	**Value**	**Obs**
0	998	1338.90	221
0	997	1415.49	58
0	989	1627.62	147
0	978	3535.44	414
0	975	4522.68	415

We want to put comments in after the first procedure.

The FREQ Procedure

SITE	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	1000	100.00	1000	100.00

We also want comments after the second procedure is run.

(Continued Additional HTML Parmeters)
6) **NEWFILE=** parameter: In the HTML output shown above, you might have wanted a separate page (file) for each table, rather than having all tables on the same page. For this purpose, you can use NEWFILE= to specify the starting point for each new BODY file. The syntax for this option is,

```
newfile = <starting point>;
```

where a starting point can be:

NONE	- write all output to the BODY file that is currently open
OUTPUT	- start a new BODY file for each output object
PAGE	- start a new BODY file for each page of output
PROC	- start a new BODY file for each new procedure
BYGROUP	– start a new BODY file for each new bygroup.

Just as ODS "increments" the name of the anchor, ODS will also automatically increment the names of the new files. For example, if the original BODY file is named RESULTS, each new BODY file that is created based on the NEWFILE parameter will be called RESULTS1, RESULTS2, etc.

7) **PAGE=** parameter: If the NEWFILE= parameter is specified, you may also want to include the PAGE= parameter in your HTML statements:

```
page=<file-specification> ;
```

The file specified will contain a description of each page of the BODY file as well as links to the BODY files.

EXAMPLE 2, Putting it all together (again). The following program illustrates the NEWFILE= parameter and the PAGE= parameter, as well as some HTML options already discussed.

```
   libname health 'C:/Data';
1) ods listing close;
2) ods html
       path    ='C:/Data'
       (url    ='http://www.unc.edu/~jismith')
       file    ='file.htm'
       contents='contents.htm'
       frame   ='frame.htm'
       page    ='page.htm' (no_top_matter)
       newfile =page;
3) ods html select Moments;
   proc univariate data=health.meddat;
      var dentdol drugdol ;
   run;
   proc print data=health.meddat;
      var site person contyr;
   run;
4) ods html close;
```

(1) The first ODS statement closes LISTING as a destination for output. This is done to conserve resources

(2) The following things happen in this ODS statement:

- the PATH= specifies where to store your HTML files;
- the URL= sub-parameter tells ODS to use this URL for links and references;
- the FILE= tells ODS the location of the body file;
- the CONTENTS= tells ODS to use this file for links to the body file for every HTML table that is created in a PROC or DATA step.
- the FRAME= parameter puts all files included in the ODS HTML statement on one screen.
- the PAGE= parameter tells ODS to use this file to store links to the BODY file for every page of HTML that ODS creates from a PROC or DATA step.
- the NEWFILE= option tells ODS to create a new BODY file for each new page of output. In this case this would mean a new file of output for each variable in the UNIVARIATE procedure and a new file for the PROC PRINT output. The name of each new file is based on the name specified by FILE= option. The BODY files that are created in this example are FILE.HTM, FILE1.HTM, and FILE2.HTM.

(3) This ODS statement instructs ODS to send only the 'Moments' statistics from the PROC UNIVARIATE to the HTML output destination.

(4) The final ODS statement closes the HTML destination in order for output to be sent there.

Results of EXAMPLE 2

All files created from the above example are shown on the next page. The Table of Contents file comes from the CONTENTS= 'CONTENTS.HTM' parameter. Each of the references under the procedure titles is a hypertext link to the location of the respective table in the BODY file.

Below the Table of Contents file is the Table of Pages file created from the PAGE= 'PAGE.HTM' (NO_TOP_MATTER) option. Each page reference (PAGE 1, PAGE 2, PAGE 3) is a hypertext link to that page in the BODY file.

Next to the Table of Contents file and the Table of Pages file are each of the BODY files that are created. The first BODY file that is created is called FILE.HTM, and it contains the Moments data for the variable DENTDOL. This page is created first because DENTDOL is the first variable listed in the PROC UNIVARIATE, and PROC UNIVARIATE is the first PROC in the program. The second BODY file created is called FILE2.HTM and it contains the Moments data for DRUGDOL. The last page, FILE3.HTM is from the PROC PRINT (note: not all observations are included in order to conserve space).

By clicking on a reference on either the Table of Contents display or the Page file display, we can link to each of the BODY files. The Frame page, FRAME.HTM, combines the PAGE file, CONTENTS file, and whichever BODY file you create, onto one page.

We have reviewed a few of the parameters used with the HTML destination. Others can be found in the "The Complete Guide to the SAS® Output Delivery System, Version 8."

ODS OUTPUT DESTINATION

The ODS OUTPUT statement is used to specify an action, or to create one or more data sets. When you first start SAS Version 7 or Version 8, the Output destination is closed and the exclusion list is set to EXCLUDE ALL. You can change these default actions with the ODS OUTPUT statement.

Specifying an action

When used to specify an action, the syntax of the ODS OUTPUT statement is

```
ods output <action>;
```

where the action choices are

CLEAR	- set the list for the OUTPUT destination to EXCLUDE ALL.
SHOW	- display the selection or exclusion list that apply at this point in the program in the SAS log
CLOSE	- close the OUTPUT destination. Once the destination is closed, you cannot send output to this destination.

Creating output data sets

You open the Output destination by specifying the data set(s) that you would like created. To create a single output data set, the syntax is,

```
ods output <output-object> = <sas data set>;
```

where the output-object can be identified with the use of the ODS TRACE statement. This example illustrates creating an Output file:

```
 ods listing;

ods trace on / listing;
ods output BasicMeasures = measures;

proc univariate data = meddat;
      var meddol suppdol;
run;
ods trace off;
```

Here we have both the Listing destination and the Output destination open.. Although all the PROC UNIVARIATE statistics will be sent to the Listing destination (whose default value is SELECT ALL), only the BasicMeasures statistics will be sent to the Output destination. You can look in the log or in the SAS Explorer window (in this case in the WORK library), to see that ODS has created the MEASURES data set. This newly created SAS data set will have the BasicMeasures statistics for both the MEDDOL and SUPPDOL variables. By looking in the Results window and clicking on BasicMeasures, you will see in the Output window the BasicMeasures statistics. Unlike the HTML destination, you do not have to close the Output destination to have objects sent there.

To create a separate output data set for each variable used in a procedure or data step, use the following syntax:

```
ods output <output-object> (match_all)
           = <sas data set>;
```

For example, the following code will select the OneWayFreqs statistics from the PROC FREQ. The OUTPUT statement creates a different data set for each variable in the PROC FREQ procedure because of the MATCH_ALL option, and bases the name of these data sets on the name STATS:

```
ods output onewayfreqs (match_all) = stats;
proc freq data='A:/test';
run;
ods output close;
run;
```

When this program is run, the data sets created are STATS, STATS1, STATS2 ... STATSN for however many variables there are in the data set TEST. You may find, however, that you would like to combine the data sets for each variable into one data set. This is easily done.

contents.htm:

Table of Contents

- The Univariate Procedure
  - DENTDOL
  - Moments
  - DRUGDOL
  - Moments
- The Print Procedure
  - Data Set IN.AGGREC

page.htm:

Table of Pages

- The Univariate Procedure
  - Page 1
  - Page 2
- The Print Procedure
  - Page 3

file.htm:

The UNIVARIATE Procedure
Variable: DENTDOL

Moments			
N	100	**Sum Weights**	100
Mean	52.38	**Sum Observations**	5238
Std Deviation	226.742535	**Variance**	51412.1774
Skewness	8.17753631	**Kurtosis**	73.6960078
Uncorrected SS	5364172	**Corrected SS**	5089805.56
Coeff Variation	432.879984	**Std Error Mean**	22.6742535

file2.htm:

The UNIVARIATE Procedure
Variable: DRUGDOL

Moments			
N	100	**Sum Weights**	100
Mean	14.8721	**Sum Observations**	1487.21
Std Deviation	34.3546782	**Variance**	1180.24391
Skewness	3.32771071	**Kurtosis**	13.1047585
Uncorrected SS	138962.083	**Corrected SS**	116844.147
Coeff Variation	231.000855	**Std Error Mean**	3.43546782

file3.htm:

Obs	SITE	PERSON	CONTYR
1	1	MA250247	01
2	1	MA250247	02
3	1	MA250247	03
4	1	MA250247	04
5	1	MA250247	05
6	1	MA250255	01
7	1	MA250255	02
8	1	MA250255	03
9	1	MA250255	04
10	1	MA250255	05
11	1	MA250263	01
.	.	.	.
.	.	.	.
.	.	.	.
93	1	MA25162A	05
94	1	MA251638	01
95	1	MA251638	02
96	1	MA251638	03
97	1	MA251638	04
98	1	MA251638	05
99	1	MA251646	01
100	1	MA251646	02

First you create a macro variable, which stores a list of the data sets that are created in the ODS OUTPUT statement. In a separate DATA step, you combine the data sets by concatenation. This is illustrated in the example below.

```
ods output OneWayFreqs (match_all=name)=stats;
```

To concatenate the data sets, you specify

```
data all;
   set &name;
run;
```

A little advice about using the MATCH_ALL option. In the following program, separate data sets are created for the Moments data, but not for the Basic Measures data. All the variables are included in one data set for the Basic Measures statistics.

```
ods output BasicMeasures Moments
           (match_all)=moments;
```

To create output separate output data sets for both sets of statistics, we would need to specify:

```
ods output BasicMeasures (match_all)= measures
           Moments (match_all) = moments;
```

To create a permanent data set with the OUTPUT statement, use the following syntax:

```
libname in 'A:/';
ods output BasicMeasures = in.measures;
```

OUTPUT Parameters

PERSIST parameter: This parameter is useful if you are creating output data set and want the data set definition to endure even when the procedure or DATA step ends, until you explicitly modify the list. The syntax is:

```
ods output
output-object<(MATCH_ALL<=macro-var-name>
PERSIST=PROC|RUN)>=<SAS-data-set> ;
```

The PERSIST parameter specifies when to close any data sets that are being created, and when to remove output objects from the SELECTION list for the OUTPUT destination. The PERSIST parameter can only be used in conjunction with the MATCH_ALL parameter.

PROC argument: The PROC argument to the PERSIST parameter preserves the list of definitions that are specified in the ODS OUTPUT statement across step boundaries. This means that the list of output objects specified in the ODS OUTPUT statement is preserved even after the procedures or DATA steps have completed. You must explicitly modify the list to change the definitions; e.g. with

```
ods output exclude all;
```

RUN argument: The RUN argument to the PERSIST parameter serves exactly the same function as the PROC statement, as it also keeps the data sets open. The following is an example of a program that implicitly uses the RUN argument but does not specify the PERSIST option (although it was intended to).

```
ods output OneWayFreqs(MATCH_ALL=name)=stats;
proc freq data=health.test;
run;
proc freq data=health.test2;
run;
```

In this case, the data sets are not created for the second PROC FREQ. Without the PERSIST option, the second procedure is treated as a step boundary, and a data set for the variables in HEALTH.TEST2 is not created. This is easily corrected by explicitly specifying:

```
ods output OneWayFreqs(match_all=name
           persist=proc) = stats;
```

CONCLUSION

The purpose of this paper was to help you get started using the Output Delivery System. Our examples illustrate that you can create web pages with the addition of as little as one line of code to your existing SAS program. With the addition of a single OUTPUT statement you can also create one or more SAS data sets. With a few additional words you can select your object objects and send them to more than one destination at a time. The best way to convince yourself, though, is to visit our website and submit the examples presented in this paper for yourself. The website can be found at:

http:// www.unc.edu/~lkbryant/odsworkshop

REFERENCES

SAS® Version 8 Software.

SAS® Institute, The Complete Guide to the SAS® Output Delivery System, Version 8

ACKNOWLEDGEMENTS

We wish to thank Paul Kent, Base SAS R&D Director, and Chris Olinger, Base SAS Software Manager, for teaching classes on these subjects at our UNC site and at local SAS users group meetings. We also wish to thank the SAS Technical Support Division for answering our questions so promptly regarding items in this paper.

CONTACT INFORMATION

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

Lara K. Bryant
Jordan Institute for Families
CB 3550
301 Pittsboro St.
Chapel Hill, NC 27599
Email: lbryant@email.unc.edu

Sally S. Muller
Jordan Institute for Families
CB 3550
301 Pittsboro St.
Chapel Hill, NC 27599
Email: sally@email.unc.edu
Work: 919-843-7798
Fax: 919-967-7015

Ray Pass
Ray Pass Consulting
5 Sinclair Place
Hartsdale, NY 10530
Email: raypass@att.net
Work: 914-693-5553
eFax: 914-206-3780