Paper S1-12-2013

Data Presentation 101: An Analyst's Perspective

Deanna Chyn, University of Michigan, Ann Arbor, MI
Anca Tilea, University of Michigan, Ann Arbor, MI

ABSTRACT

You are done with the tedious task of data cleaning, and now the fun begins. Your analysis has produced several results, which you must present in a useful manner to a statistician (if you are lucky) or a non-statistician (as is usually the case). SAS® provides a multitude of resources to do this very thing: the HISTOGRAM statement in PROC UNIVARIATE and/or PROC SGPLOT, PROC REPORT, Output Delivery System (ODS) statements for various PROCs, and so on. This paper will describe a few of these methods that we, as data analysts, have found to be most helpful when presenting results. This paper is intended for the novice SAS® user with basic to intermediate skills and who is using SAS® 9 or above.

INTRODUCTION

You have cleaned and analyzed your data, and now you need to present results. Although SAS® produces output (i.e., the .LST file), it is not always the ideal means of displaying your results. The question is, how do you transform the .LST file into a formatted table without doing too much manual work? SAS® does more than simply cleaning and analyzing data; SAS® can tailor output to your needs.

GETTING STARTED

1. ODS TRACE ON/OFF

The ODS TRACE ON/OFF command, used with any SAS® PROC, creates a list in the Log window (i.e., the .LOG file) of all available SAS® data tables that the procedure is able to generate.

For example, running the code below:

```
ODS TRACE ON;
PROC REG DATA = SASHELP.CLASS;
       MODEL WEIGHT = AGE;
RUN;
ODS TRACE OFF;
```

will produce the following .LOG file:

```
Output Added:
-------------
Name:       NObs
Label:      Number of Observations
Template:   Stat.Reg.NObs
Path:       Reg.MODEL1.Fit.Weight.NObs
-------------
```

```
Output Added:
-------------
Name:       ANOVA
Label:      Analysis of Variance
Template:   Stat.REG.ANOVA
Path:       Reg.MODEL1.Fit.Weight.ANOVA
-------------

Output Added:
-------------
Name:       FitStatistics
Label:      Fit Statistics
Template:   Stat.REG.FitStatistics
Path:       Reg.MODEL1.Fit.Weight.FitStatistics
-------------

Output Added:
-------------
Name:       ParameterEstimates
Label:      Parameter Estimates
Template:   Stat.REG.ParameterEstimates
Path:       Reg.MODEL1.Fit.Weight.ParameterEstimates
```

What does this tell you? It gives you the names of all possible SAS® data sets that PROC REG can create by using the ODS statement. For example, if you want to create a SAS® data set named "MY_EST" that contains the estimates from your model, add the following statement (in bold) to your procedure:

```
PROC REG DATA = SASHELP.CLASS;
  MODEL WEIGHT = AGE HEIGHT;
  ODS OUTPUT PARAMETERESTIMATES = MY_EST;
RUN;QUIT;
```

You now have a SAS® data set in the WORK library that can be manipulated as needed. All SAS® procedures have the capability of generating such data tables; just add the ODS TRACE ON/OFF options to see what these are. Keep in mind that when using the ODS statement with a PROC, you cannot use the NOPRINT option in the procedure statement; this will suppress the ODS data set from being created.

2. ODS TAGSETS.EXCELXP

Now you have a nice, clean data set from your analysis—all labels are in place, and all values are beautifully formatted. You are ready to export this data set to Excel, which you do via PROC EXPORT (or the EXPORT Wizard). But when you open the file, something is amiss. What are all these 1s and 0s doing here? You made the formats in SAS®; you can see them in your SAS® data set. Excel, it seems, is not ready to accept your formats.

Let us introduce you to the magnificent (and our favorite) ODS TAGSETS.EXCELXP.

Let's say you want to present the parameter estimates from a simple linear regression, and you format the p-value as significant or not. You used a PROC FORMAT to make significant values display as "SIGNIFICANT" instead of their numeric value.

```
PROC FORMAT;
     VALUE P_VAL LOW-0.05 = "SIGNIFICANT"
                  0.05 - HIGH = "NOT SIGN."
     ;
QUIT;
```

Now, instead of using PROC EXPORT (or the EXPORT Wizard), you should use ODS TAGSETS.EXCELXP to retain the desired formats on the output file:

```
ODS TAGSETS.EXCELXP FILE = "[YOUR_PATH]\MY_FORMATTED_FILE.XLS"
STYLE = MINIMAL;
    PROC PRINT DATA = MY_EST;
       VAR Variable Estimate StdErr Probt;
       FORMAT PROBT P_VAL.;
     RUN;
ODS TAGSETS.EXCELXP CLOSE;
```

As you can see in the table below, the format associated with the p-value is now displayed.

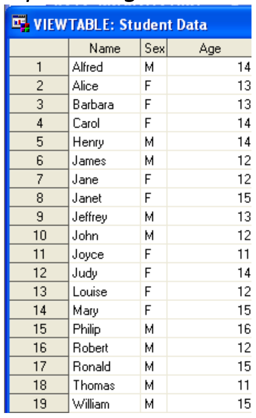
Obs	Variable	Estimate	StdErr	Probt
1	Intercept	-141.22376	33.38309	**SIGNIFICANT**
2	Age	1.27839	3.1101	**NOT SIGN.**
3	Height	3.59703	0.90546	**SIGNIFICANT**

This simple option is very flexible and can be used with any procedure that generates output (e.g., PROC PRINT, PROC MEANS with an OUTPUT OUT= statement, PROC REPORT). There are also many options available to tailor the Excel file to your liking, which are specified in an OPTIONS statement in the ODS TAGSETS.EXCELXP statement: freezing the top row, setting the column widths, naming the individual spreadsheets, etc.

3. PROC TRANSPOSE WITH ID STATEMENT

When you need to transpose data using PROC TRANSPOSE, the basic syntax does not necessarily produce the most readable output. When transposing the data from long…

Input: Long



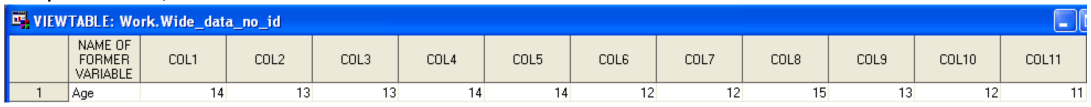
VIEWTABLE: Student Data

	Name	Sex	Age
1	Alfred	M	14
2	Alice	F	13
3	Barbara	F	13
4	Carol	F	14
5	Henry	M	14
6	James	M	12
7	Jane	F	12
8	Janet	F	15
9	Jeffrey	M	13
10	John	M	12
11	Joyce	F	11
12	Judy	F	14
13	Louise	F	12
14	Mary	F	15
15	Philip	M	16
16	Robert	M	12
17	Ronald	M	15
18	Thomas	M	11
19	William	M	15

…to wide, adding a simple statement called "ID" comes in handy.

```
PROC SORT DATA = SASHELP.CLASS OUT = CLASS_SORTED;
   BY NAME;
      PROC TRANSPOSE DATA = CLASS_SORTED OUT = WIDE_DATA;
           VAR AGE;
RUN;
```

Output: Wide, without the ID statement



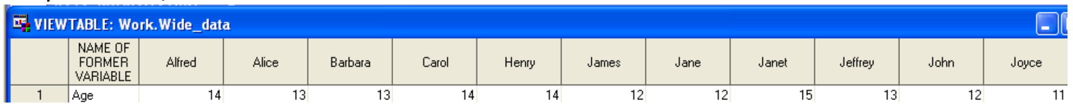
VIEWTABLE: Work.Wide_data_no_id

	NAME OF FORMER VARIABLE	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9	COL10	COL11
1	Age	14	13	13	14	14	12	12	15	13	12	11

As you can see, the variable name *COL*n is not very helpful when you want to associate the age with a particular person in the data set. So you add the ID statement:

```
PROC TRANSPOSE DATA = CLASS_SORTED OUT = WIDE_DATA;
            ID NAME;
            VAR AGE;
RUN;
```

Output: Wide, with ID statement



VIEWTABLE: Work.Wide_data

	NAME OF FORMER VARIABLE	Alfred	Alice	Barbara	Carol	Henry	James	Jane	Janet	Jeffrey	John	Joyce
1	Age	14	13	13	14	14	12	12	15	13	12	11

The resulting output is now in a more readable format.

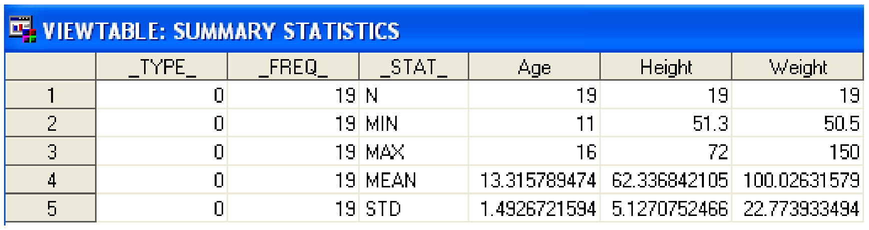
4. OUTPUT OUT= STATEMENT FOR PROC MEANS

As an analyst, one of your most common tasks is running a PROC MEANS on continuous variables. You look in the Output window (or the .LST file) to examine the results; everything is aligned and easily readable:

```
                         The MEANS Procedure

 Variable    N            Mean         Std Dev         Minimum         Maximum
------------------------------------------------------------------------------
Age         19      13.3157895       1.4926722      11.0000000      16.0000000
Height      19      62.3368421       5.1270752      51.3000000      72.0000000
Weight      19     100.0263158      22.7739335      50.5000000     150.0000000
------------------------------------------------------------------------------
```

You now need to present these results, so you want to create a data set that looks like the Output window content, then export that data set as an Excel file (for example). To do this, you need the OUTPUT OUT= statement in the PROC MEANS, which will create a SAS data set containing the summary statistics.

```
PROC MEANS DATA = SASHELP.CLASS;
     VAR AGE HEIGHT WEIGHT;
OUTPUT OUT = MY_MEANS;
RUN;
```

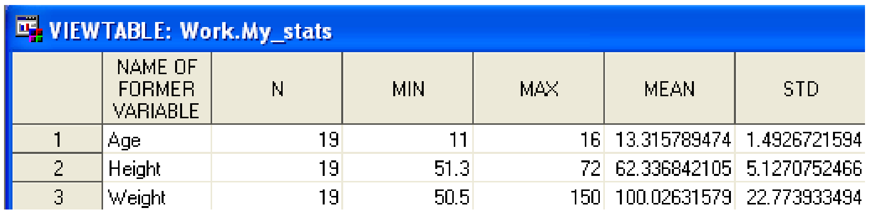


VIEWTABLE: SUMMARY STATISTICS

	TYPE	_FREQ_	_STAT_	Age	Height	Weight
1	0	19	N	19	19	19
2	0	19	MIN	11	51.3	50.5
3	0	19	MAX	16	72	150
4	0	19	MEAN	13.315789474	62.336842105	100.02631579
5	0	19	STD	1.4926721594	5.1270752466	22.773933494

But wait! The data set MY_MEANS looks nothing like the content from the Output window. Don't fret – just transpose:

```
PROC TRANSPOSE DATA = MY_MEANS OUT = MY_STATS;
ID _STAT_;
     VAR AGE HEIGHT WEIGHT;
RUN;
```



VIEWTABLE: Work.My_stats

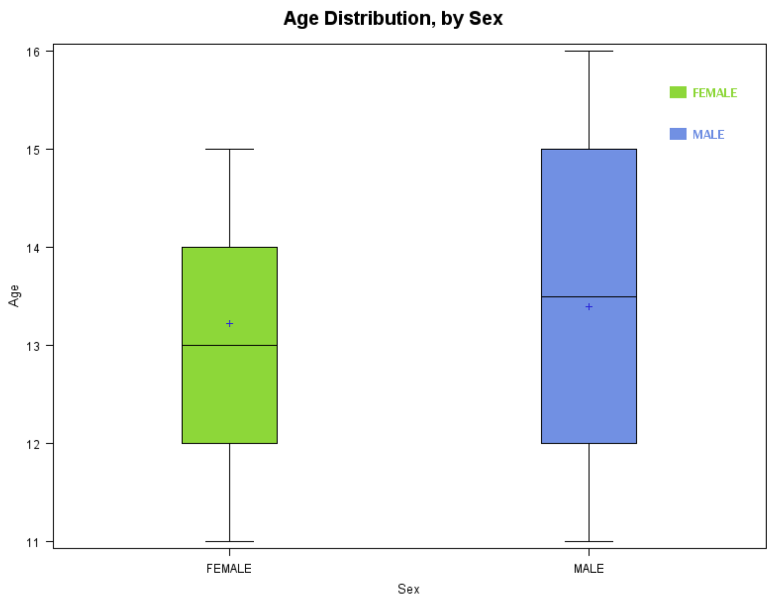
	NAME OF FORMER VARIABLE	N	MIN	MAX	MEAN	STD
1	Age	19	11	16	13.315789474	1.4926721594
2	Height	19	51.3	72	62.336842105	5.1270752466
3	Weight	19	50.5	150	100.02631579	22.773933494

5. ANNOTATE LEGEND FOR BOXPLOT

Boxplots are a commonly requested graphical display of basic statistics. If you haven't already, you will be asked to produce boxplots many times over the course of your career as an analyst. One of the most common SAS® procedures used to create a boxplot is PROC BOXPLOT; however, it does not include an option for displaying a legend. The below code demonstrates how you can annotate a legend to your boxplot, using the ANNO option in the PROC BOXPLOT statement. (Note that if you are using SAS® 9.3 or higher, the KEYLEGEND statement is available in PROC SGPLOT; the more advanced SAS® user may also want to use PROC TEMPLATE, including a DISCRETELEGEND statement, in conjunction with a PROC SGRENDER.)

```
GOPTIONS RESET = ALL;
DATA FOR_PLOT;
     SET SASHELP.CLASS;
     IF SEX = "M" THEN COLOR = "H158aaaa";
     ELSE IF SEX = "F" THEN COLOR = "H23888aa";
RUN;


DATA ANNO;
   LENGTH FUNCTION COLOR $ 8 TEXT $ 25 STYLE $ 25;
   XSYS='3'; YSYS='3';

      /* DRAW THE FIRST SQUARE */
   COLOR="H23888aa";
   FUNCTION='MOVE'; X=85; Y=87; OUTPUT;
   FUNCTION='BAR';  X=87; Y=85; STYLE='SOLID'; OUTPUT;

      /* LABEL THE FIRST SQUARE */
   FUNCTION='LABEL'; X=88; Y=86; POSITION='6';
```

```
            STYLE="'ALBANY AMT/BOLD'"; SIZE=1; TEXT='FEMALE'; OUTPUT;

            /* DRAW THE SECOND SQUARE */
         COLOR= "H158aaaa";STYLE='SOLID';
         FUNCTION='MOVE'; X=85; Y=80; OUTPUT;
      FUNCTION='BAR';  X=87; Y=78; STYLE='SOLID'; OUTPUT;

            /* LABEL THE FIRST SQUARE */
      FUNCTION='LABEL'; X=88; Y=79; POSITION='6';
      STYLE="'ALBANY AMT/BOLD'"; ; SIZE=1; TEXT='MALE'; OUTPUT;
      RUN;

      PROC SORT DATA = FOR_PLOT;BY SEX;RUN;
      PROC FORMAT;
          VALUE $ GENDER "M" = "MALE" "F" = "FEMALE";
      QUIT;
      FILENAME PLOT "&PATH.\Output\BOXPLOT_ANNO.PNG";
      GOPTIONS RESET = ALL DEVICE = PNG GSFNAME = PLOT GSFMODE = REPLACE;
      PROC BOXPLOT DATA = FOR_PLOT ANNO = ANNO;
         PLOT AGE*SEX/BOXSTYLE = SCHEMATIC BOXWIDTH = 12 CBOXFILL = (COLOR);
         FORMAT SEX $GENDER.;
           TITLE "Age Distribution, by Sex";
      RUN;QUIT;
```



6. PROC SQL ON DICTIONARY.COLUMNS

There are times when you are given a data set that is labeled beautifully but whose variable names are very long and/or are not intuitive. For example, the variable for a patient's age is labeled "Age" although the actual variable name is *person_age_to_date*. Wouldn't it be better, perhaps for display purposes, for the variable to be named *age*? Of course it would.

To rename variables with their associated labels, you can make use of some basic information—such as variable name, type, label and format—that is stored not in the data set itself, but rather in a handy table called DICTIONARY.COLUMNS. This is a secondary data set that SAS® creates for each data set you work with. We tap into this information via PROC SQL. In the example below, you use the SAS® data set SAS®HELP.SHOES to rename its variables (stored in the column NAME) with their associated labels (stored in the column LABEL). You are creating a macro variable (called &NEW_LABEL) that takes the form NAME = LABEL, for all the values in the NAME column.

```
PROC SQL;
 SELECT CATT(NAME, "=", COMPRESS(LABEL))
 INTO: NEW_LABEL SEPARATED BY " "
 FROM DICTIONARY.COLUMNS
 WHERE LIBNAME = "SASHELP" & MEMNAME = "SHOES" & LABEL NE "";
QUIT;

%PUT &NEW_LABEL;
       Stores=NumberofStores Sales=TotalSales Inventory=TotalInventory
Returns=TotalReturns
```

You have the information stored in the macro variable; now, you need to use it. The data step below renames the variables in the data set to their corresponding labels:

```
DATA WANT;
     SET SASHELP.SHOES;
     RENAME &NEW_LABEL.;
RUN;
```

7. %NRSTR FUNCTION

Let's say you're asked to put a label on a report or a graphic that contains text of the form, "%Males" or "Beautiful&Smart". You notice that these labels contain special characters but do not contain spaces—this is a problem for SAS®, which interprets these values as macro calls (or macro variables) and produces an ERROR message in the LOG file. To avoid this situation, you must use the macro function %NRSTR, which will ignore the special characters by treating them as regular text. Please note the use of single quotes; double quotes will work as intended but will create a WARNING message in the LOG file.

```
ODS GRAPHICS ON;
%LET text1 = %NRSTR(%things);
%LET text2 = %NRSTR(some label  &beautiful words);
PROC SGPLOT DATA = SASHELP.CLASS;
   SCATTER X = height Y = weight;
     XAXIS LABEL = '&text1.';
     YAXIS LABEL= '&text2.';
RUN;
```

CONCLUSION

To minimize the amount of time an analyst spends on formatting and reformatting (and reformatting…) her results, we suggest using a few of these tricks to expedite the process. Using basic PROC SQL or ODS statements may make all the difference in time spent cleaning up and clarifying analytic output.

CONTACT INFORMATION

Deanna Chyn
University of Michigan, School of Public Health, Department of Biostatistics
1415 Washington Heights, Ste 3645A (SPH I)
Ann Arbor MI 48109
chyndl@med.umich.edu