

Spring 2007 OASUS Q&A

Question #1 How do you reference a list of variables for which the names end with the same 2 or 3 letters?

It is easy to create variable lists in SAS using special notation that involves either a prefix or that is based on the order of the variable in the program data vector (memory area in SAS where the variables are stored). In that regard, the following notations are possible:

- *A1-A5* : to represent variables based on a numbered range (in this case A1, A2, A3, A4 and A5) if they are present in the program data vector.
- *A:* : to represent all variables in the program data vector that start with A.
- *YHN--BHG* : to represent all variables in the order they are stored in the program data vector.
- *YHN-character—BHG* : to represent all character variables in the order they are stored in the program data vector.
- *YHN-numeric—BHG* : to represent all numeric variables in the order they are stored in the program data vector.

The following examples illustrate some of these different notations:

```
/* Variable lists based on a prefix */
DATA INPUT1;
  DROP A3-A5; /* drop A3, A4 and A5 */
  DROP X:;    /* drop all variable that start with x */

  INPUT A1 A2 A3 A4 A5 X1 X2 X3 X4 X5;
  DATALINES;
1 2 3 4 5 6 7 8 9 10
;
RUN;

/* Variable list based on the position of variables in the
program data vector */
DATA INPUT2;
  DROP C14 -- U19; /* Drop C14,D2,Z90, U19 */
  INPUT A11 BOO6 C14 D2 Z90 U19 R16 M8 A3 E77;
  DATALINES;
1 2 3 4 5 6 7 8 9 10
;
RUN;
```

In the case of variable lists based on a suffix, SAS does not provide any special syntactic notation. One must create a macro variable that will be populated using the internal SAS data dictionary or indirectly via PROC CONTENTS.

One point to raise is that in an SQL statement it is not possible to create a variable list using a special notation simply because in PROC SQL there is no program data vector. One has to use macro processing for any type of variable list. Here is an example of creating a variable list using the macro facility in PROC SQL.

```
/* Variable list created using the macro facility */
DATA INPUT3;
    INPUT A1B A2B A3 A4 A5 X1 X2 X3B X4B X5B;
DATALINES;
1 2 3 4 5 6 7 8 9 10
;
RUN;

PROC SQL ;
    SELECT COMPRESS(NAME) as VAR_LIST INTO :VAR_LIST
    SEPARATED BY ","
    FROM DICTIONARY.COLUMNS
    WHERE LIBNAME = 'WORK' AND MEMNAME = 'INPUT3'
    AND NAME LIKE "%B";

    SELECT &VAR_LIST
    FROM INPUT3;
QUIT;
```

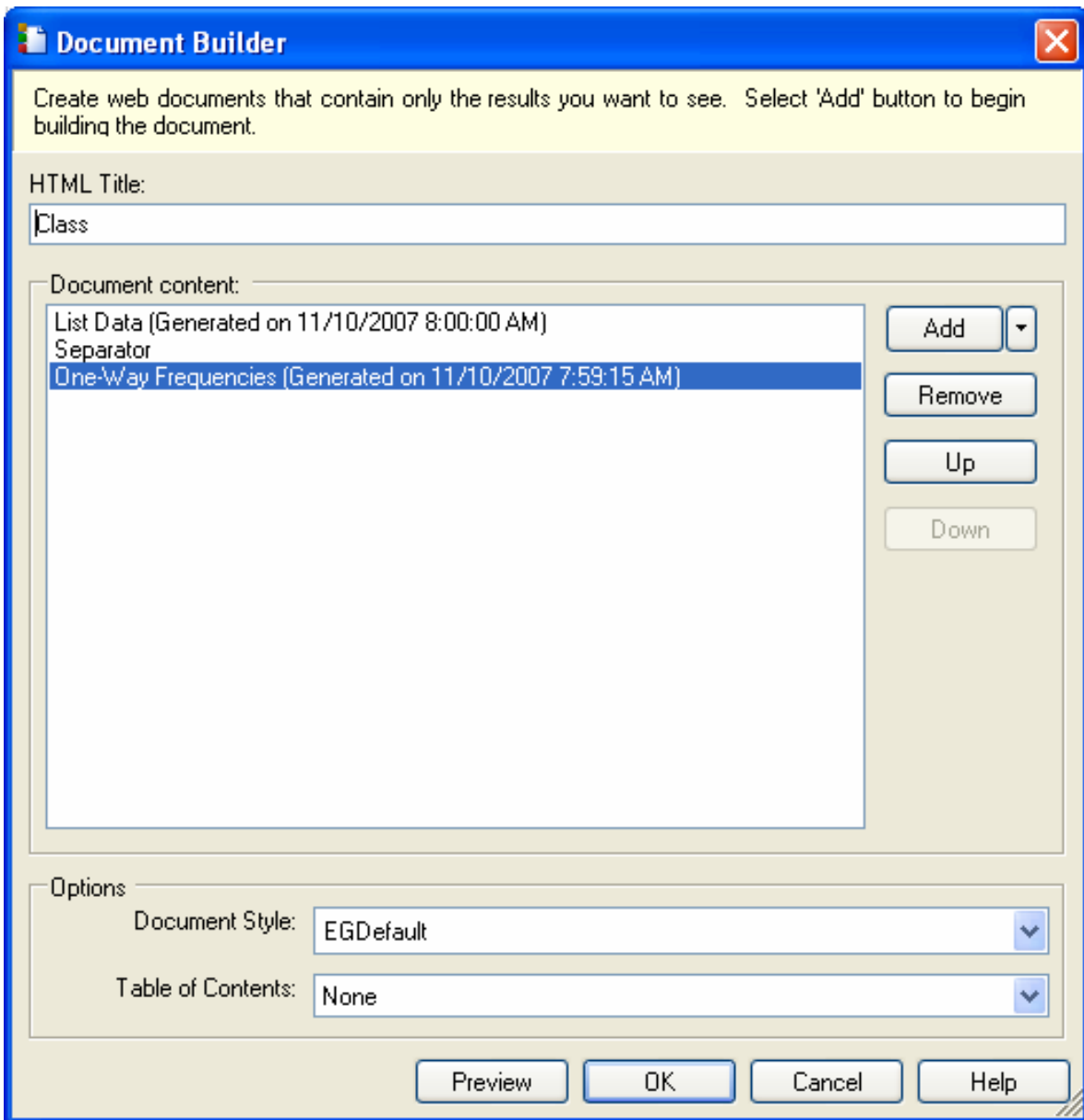
Reference

Morris, Robert J. (2005), "Text Utility Macros for Manipulating Lists of Variable Names", *Proceedings of the 30th Annual SAS Users Group International Conference*, 029-30.

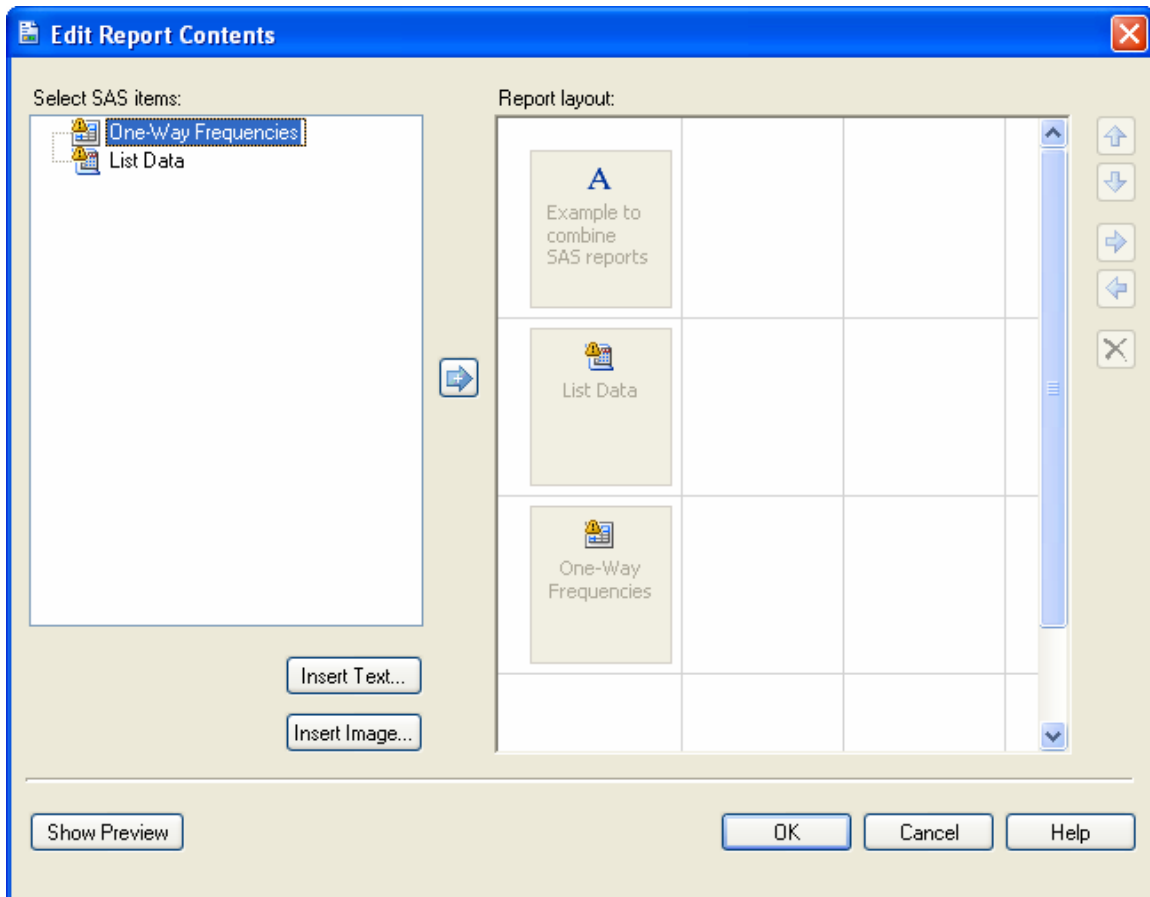
Pollack, Stuart (2005), "Techniques for Effectively Selecting Groups of Variables", *Proceedings of the 30th Annual SAS Users Group International Conference*, 057-30

Question #2 In EG, how do you output the results of several tasks into a single RTF file?

Before exploring the RTF dilemma, it is important to note that there are 2 facilities available in EG to create composite documents. The first one is the Document Builder that enables you to combine the HTML results from multiple tasks in your project into a single document. It is accessible from the Tools menu and the following is a screenshot of this facility.



The second facility is the newly available in EG 4 SAS report facility. This facility allows you to create results in the SAS Report format rather than RTF or HTML or PDF. You can then use those results to create a customized report (option File/New/Report) that you can print, export, and share with other SAS applications. You can add multiple results to the report along with text and images, and you can choose how to arrange them in the report. When the task or code that generated the results is rerun, the results in the report are automatically refreshed. Here is a screenshot of this new facility.



Now let's address the RTF dilemma. The very nature of the RTF format makes it impossible to combine RTF files "after the fact". However, the ODS facility allows you to keep an RTF destination open and run many tasks before closing it. All output generated by the tasks will be stored in a single RTF file. Unfortunately, it is only possible to do it in EG using code. An EG task will always close all ODS destinations once it completes. Exporting coding code from many tasks is easy however and once packaged into one code node, the results can be set-up so that they are directed to one RTF file by setting the results accordingly (using Properties/Results).

Question #3 How to set the line (to double-line for example) that separate rows in a table created by the TABULATE procedure? The final output is either in HTML or PDF

Like most procedures in SAS, PROC TABULATE is capable of producing output to various destinations such as HTML, PDF or RTF via the Output Delivery System (ODS). When you select a destination that is capable of formatting beyond monospace characters, you can set the style element that the procedure uses for various parts of the table. Style elements determine presentation attributes, such as font face, font weight, color, and so forth.

Style elements can be overridden with style attributes. This allows the user to set some attributes for very specific part of a table. For example, it is possible to change the foreground or the background of data cells. To change the line that separates the rows, one can attempt to change the border using the BORDERWIDTH attribute. Unfortunately, this attribute formats the entire border of the cell (not just the upper and lower border). The example below shows how to change the border for all data cells and variables:

```

PROC TABULATE
  DATA=SASHELP.CLASS STYLE={BORDERWIDTH=.25CM};
  VAR WEIGHT HEIGHT /      STYLE={BORDERWIDTH=.25CM};
  KEYWORD MEAN /          STYLE={BORDERWIDTH=.25CM} ;
  CLASS SEX /             ORDER=UNFORMATTED MISSING;
  CLASS AGE /             ORDER=UNFORMATTED MISSING;
  TABLE
    /* Row Dimension */
    WEIGHT*MEAN
    HEIGHT*MEAN,
    /* Column Dimension */
    SEX
    AGE
  ;
RUN;
QUIT;

```

		Sex		Age					
		F	M	11	12	13	14	15	16
Weight	Mean	90.11	108.95	67.75	94.40	88.67	101.88	117.38	150.00
Height	Mean	60.59	63.91	54.40	59.44	61.43	64.90	65.63	72.00

Another possibility is the RULES attribute. This controls line drawing. The possible values are ALL, COLS, GROUPS, NONE, and ROWS. None of these choices produce double-lines but they do provide a fair amount of flexibility as shown in the following example:

```

PROC TABULATE DATA=SASHELP.CLASS;
  CLASS AGE SEX;
  VAR HEIGHT;
  TABLES AGE, SEX*HEIGHT*SUM/
           STYLE={ BORDERCOLOR=CX000000
                   FRAME=BOX
                   RULES=ROWS
                   CELLSPACING=0};
RUN;

```

		Sex	
		F	M
Height		Height	Height
Sum		Sum	Sum
Age			
11	51.30	57.50	
12	116.10	181.10	
13	121.80	62.50	
14	127.10	132.50	
15	129.00	133.50	
16	.	72.00	

Finally, a very interesting approach is to alternate the color using SAS formats to improve readability. The following example illustrates this approach. A user-defined format called abc is first created and is used to map values to two different colours. This format is then invoked in the call to PROC tabulate to dynamically derive the value to set the background style attribute for age.

```

PROC FORMAT;
  VALUE ABC      11,13,15 = WHITE
                12,14,16 = YELLOW;
PROC TABULATE DATA=SASHELP.CLASS;
  VAR HEIGHT;
  CLASS AGE;
  CLASSLEV AGE /STYLE={BACKGROUND=ABC.};
  TABLE AGE*{STYLE=<PARENT>},HEIGHT*MEAN;
RUN;

```

	Height
	Mean
Age	
11	54.40
12	59.44
13	61.43
14	64.90
15	65.63
16	72.00

Question #4 How to merge tables created by the TABULATE procedure and with the same columns but different statistics?

When using PROC TABULATE, a table is generated with all the statistics requested for every category created using the classification variables. A technique to customize the statistics depending on the category is to first invoke TABULATE to generate all the statistics to an output dataset and then use this output dataset as input to a second pass to PROC TABULATE. In the second pass, PROC TABULATE is invoked once per category and depending on the category a specific set of statistics is requested. The following example uses this technique:

```
* Write all your statistics to a dataset;
FOOTNOTE;
TITLE1 "ALL AGES ALL STATISTICS";
PROC TABULATE DATA=SASHELP.CLASS OUT=WORK.STATS;
VAR HEIGHT;
CLASS SEX AGE;
TABLE AGE*HEIGHT*(SUM MEAN MIN N),SEX;
RUN;

* Read the summary dataset multiple times to produce
multiple reports;
%LET AGE=11;
TITLE1 "AGE &AGE";
PROC TABULATE DATA=WORK.STATS(WHERE= (AGE= &AGE));
VAR HEIGHT_SUM HEIGHT_MEAN HEIGHT_N;
CLASS SEX AGE;
LABEL HEIGHT_SUM = 'SUM'
      HEIGHT_MEAN = 'MEAN'
      HEIGHT_N = 'N' ;
KEYLABEL SUM = ' ';
TABLE AGE*(HEIGHT_SUM HEIGHT_MEAN HEIGHT_N),SEX;
RUN;

%LET AGE=12;
TITLE1 "AGE &AGE";
PROC TABULATE DATA=WORK.STATS(WHERE= (AGE= &AGE));
VAR HEIGHT_SUM HEIGHT_MEAN HEIGHT_N;
CLASS SEX AGE;
LABEL HEIGHT_MEAN = 'MEAN' ;
KEYLABEL SUM = ' ';
TABLE AGE*(HEIGHT_MEAN),SEX;
RUN;
```

The result is as followed:

AGE 11

		Sex	
		F	M
Age			
11	sum	51.30	57.50
	mean	51.30	57.50
	n	1.00	1.00

AGE 12

		Sex	
		F	M
Age			
12	mean	58.05	60.37

It is worth noting that it is possible to use other SAS procedures in the second pass to generate reports to the desire format. A good candidate would be PROC REPORT. Even the DATA step in combination with ODS could be used to that effect to meet very specific requirements.

Question #5 How to save plots with specific names? Such as plot&grp1, plot&grp2?

Whenever you run a SAS/GRAPH procedure, output graphics are always saved in a graphics catalog. By default, a temporary graphics catalog named WORK.GSEG is created in the work library. An additional temporary or non-temporary catalog can be created by using the GOUT= option on the PROC statement that invokes the graphics procedure. Graphics entries are appended to an existing catalog by default but this behaviour can be altered by using the GOUTMODE= option on a GOPTIONS statement (APPEND or REPLACE).

Entries in a graphics catalog always have a name and a description. Default names and descriptions are derived from the graphics procedure and variables used to produce a catalog entry. The default name is the procedure name, with a numeric suffix used if more than one graph is produced in a session. For example, in a PROC GPLOT session, the first or only entry would be named GPLOT and any subsequent entries would be named GPLOT1, GPLOT2, ... GPLOTn. The default description is derived from the procedure statement that requests the graph. For example, for a PLOT statement of the form PLOT y-variable*x-variable, the default entry description would be PLOT OF y-variable*x-variable.

You can assign a name of your choice to a catalog entry using the NAME='entry-name' option of the procedure statement that requests the graph. In PROC GPLOT, for example, this would be the BUBBLE or PLOT statement. The entry-name value has a maximum length of eight characters. If the coded entry-name value includes a macro variable, take care to ensure that the combined length of any hard-coded name portion plus the resolved value of the macro variable does not exceed eight characters. You can also assign an entry description value of your choice using the DESCRIPTION= option on the same statement that supports the NAME= option. The description value has a maximum length of 256 characters.

PROC GREPLAY supports the displaying of graphs saved as catalog entries. The input graphics catalog is specified by either the IGOUT option of the PROC statement or by a separate IGOUT statement. Entries can be replayed by name or by relative position or all entries can be displayed by coding REPLAY _ALL_;

TIP: If graphics catalog entries are created under the control of one device driver and then replayed under the control of a different device driver, distortion can occur in the replayed graphs because of differences in the characteristics of device drivers, particularly regarding aspect ratio. For example, if graphics catalog entries are created in a Windows workstation session using the default device driver, which supports your monitor, and then the entries are replayed using the device driver for the grey-scale version of the Windows printer driver, WINPRTG, the printed graphs will be slightly stretched. This will be particularly noticeable with pie charts or bubbles. Distortion can be avoided by using the same device driver when creating the catalog entries as will be

used when replaying the graphs. The device driver is selected using the GDEVICE option of the GOPTIONS statement.

The following is an example using sashelp.class as an input data set:

```

/* Set graph options */
TITLE1 'SASHELP.CLASS';
TITLE2 'PLOT OF AGE BY HEIGHT';

/* Run gplot and save with an entry name specified by &gplot1 */
%LET PLOTNAME=MYPLOT1;
PROC GLOT DATA=SASHELP.CLASS;
    PLOT AGE*HEIGHT / NAME="&PLOTNAME";
RUN;
QUIT;

TITLE1 'CONTENTS OF GSEG AFTER GPLOT';
PROC CATALOG CATALOG=GSEG;
    CONTENTS;
RUN;

```

The result is as follows when run from Enterprise Guide with default output options:

