

## Your BASE Dollars at Work

Information Products Staff  
SAS Institute Inc.

### Abstract

The BASE procedures group (a.k.a. Information Products) at SAS Institute has been hard at work adding new and useful features to the heart of the SAS® System. This talk describes this work and uses generic "business oriented" examples to place these features in a context that demonstrates their use in everyday programming.

Covered are:

- General additions to SAS® Software that should make the programmers life an easier one.
- New SUMMARY/TABULATE options like percentiles, independent ordering of class variables, and the pre loading of the class-variable space using formats and/or a data sets of desired class level values.
- The ODS statement for producing output data sets and HTML output from all SAS procedures, High Fidelity Printing and more. ODS is an acronym for Output Delivery System.
- ODS specific additions to PROC REPORT, PROC TABULATE and the SAS/Graph® Procedures.

### General Improvements

SAS Software (finally!) permits table and column names to be up to 32 characters long. Labels may now be up to 256 characters, and the values of character variables themselves can exceed 30000 bytes.

SAS Access® Software has more direct access to the popular relational database systems – Imagine

```
LIBNAME MYDATA DB2 ...;  
PROC FREQ DATA=MYDATA.table;  
RUN;
```

To access your RDBMS data, and even

```
DATA MYDATA.newtable;  
SET ...;  
* RUN;
```

To create tables in your database (provided your DBA allows you of course – often the more difficult challenge!).

PROC SQL has been upgraded to be more DBMS alert too – it will now detect that two or more tables in a join are from the same RDBMS and construct a useful Passthru SQL query expression to pass the work of doing the join off to the data base.

We hope the combination of longer names and direct DBMS access allows your SAS programs to be more directly aligned with (for example) the element names in your corporate data model.

### A Note on Names

SAS names can be any sequence of characters. The catch is that if the characters are not the traditional alpha, digit or underscore then the SAS programmer is obliged to use a quoted-name notation borrowed from the familiar SAS date constants.

The code:

```
DATA MYDATA;  
  'My Date'v = today();  
RUN;
```

Creates a variable name with a space in it. Since SAS Syntax is so free form and most often "white space delimited" we needed to invent this '...v syntax to avoid confusing

this one variable with two others – My and Date.

SAS Software uses a *Case Preserving* model for variable names. The first mention of a variable defines the case of the letters in the variable name (much like it also defines the data type and length). Subsequent references to the same variable can use a different case.

Please accept our apologies for the endless debate that will ensue on whether VariableName or Variable\_Name is the preferred way to spell things in your company ☺. It might be interesting to know that we recognise case transitions like “eN” as a possible split point in situations where we need to “wrap” the name – for a column heading perhaps.

## ODS

Please see the paper “*ODS for Data Analysis: Output As-You-Like-It in Version 7*” for a more complete treatment of ODS features.

SAS Output has been criticized for its Green-Bar characteristics. [Green-Bar being a generic term for “traditional” computer paper mounted in a chain printer – perforations along the sides, 66 lines deep, 132 characters wide, horizontal green stripes to aid readability... The stuff legends are made of.]

Until Version 7, SAS procedures that produce printed output (i.e. destined for the SAS Listing File) have had this “page of paper” as their model when generating output. Headings are centered within the LS= values. PS= determines how many lines will print on a page. Each procedure was responsible for formatting its own output, and this is often done one line at a time, moving characters into the “print buffer” of the program.

PROC PRINTTO has had a 20+ year reign as the “solution” to capturing output that is printed by one SAS procedure, but is required as input for some other procedure. Many SAS procedures produce output data

sets with an OUTPUT Statement or an OUTxx= option, but invariably there is some printed data point that the programmer wishes he could “capture” that is not available on the output data set.

ODS gives you features like the:

- Ability to get all printed numbers into data sets with no loss of precision
- Ability to select and exclude different pieces of output, and to combine and rearrange tables.
- Ability to format with better display technologies like HTML, Postscript and PDF.

This would be a tall order if each procedure were to retain the responsibility for formatting its own output. Each procedure would have to grow the code to do all of the marvelous things mentioned above – clearly we needed a better way.

Consider this GLM code fragment that models two variables, x and y:

```
data a;
  do a = 1 to 4;
    do b = 1 to 4;
      do c = 1 to 4;
        n = int(6*ranuni(1));
        do i = 1 to n;
          y = a + b
            + rannor(1);
          x = a * b
            / rannor(3);
          output;
        end; end; end; end;

proc glm data=a;
  class a b c;
  model x y = a|b|c
    / ssl ;
run;
```

The Output from this code arranges into a hierarchy quite comfortably. *Italics* are used to show actual output objects. **Bold** is used to show “directory” objects created to make a useful structure.

In this specific GLM example, a **Data** directory is created to summarize the values found in the input data set, and an **Analysis of Variance** directory is created for the results of the analysis. A separate directory is created for each variable being modeled.

## GLM

### Data

*Class Levels*

*Number of Observations*

### Analysis of Variance

#### Dependent Variable X

*Overall ANOVA*

*Fit Statistics*

*Type I Model ANOVA*

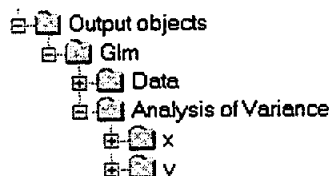
#### Dependent Variable Y

*Overall ANOVA*

*Fit Statistics*

*Type I Model ANOVA*

When run through ODS this same hierarchy is displayed using a tree control in the new SAS explorer window:



Selecting one of the directories (Y for example) shows the available tables in that directory in the right hand pane of the explorer.

| Contents of y        |      |        |
|----------------------|------|--------|
| Name                 | Size | Type   |
| Overall ANOVA        |      | Output |
| Fit Statistics       |      | Output |
| Type I Model ANOVA   |      | Output |
| Type II Model ANOVA  |      | Output |
| Type III Model ANOVA |      | Output |
| Type IV Model ANOVA  |      | Output |

ODS removes the responsibility for formatting output from the procedure completely. The procedure supplies the raw data [along with the name of the formatting instructions] to ODS, and ODS takes care of the details.

## ODS and HTML

One of the new output destinations supported by Version 7 is the format of the World Wide Web – HTML. We generate HTML that will display in “Version 3” browsers (Internet Explorer and Netscape), but can take advantage of the more modern Dynamic HTML found in IE.

The ODS Statement:

```
ODS HTML  
  file="body.html"  
  frame="frame.html"  
  contents="cont.html";
```

enables the generation of HTML formatted tables. Three separate files are mentioned.

The file referenced with frame= contains HTML that configures an HTML browser to show two panes of output. The left-hand pane is a clickable table of contents, containing hot links to the respective tables in the right-hand pane.

The file referenced with file= receives the meat of the generated HTML3.2 <table> ... </table> tag stream. This HTML shows in the right hand frame of the browser. You are free to “switch” body files without changing the contents file – this would reduce download time in situations where the viewer would not visit all the tables in the output.

The file referenced with contents= contains HTML formed from the hierarchy of output objects produced by the SAS procedures. The text is arranged to reflect the hierarchy, and is comprised of HTML links to the respective tables in the body file.

The output shown on the next page is produced by adding the ODS HTML statement to the PROC GLM statement used earlier.

SAS OUTPUT - Microsoft Internet Explorer

File Edit View Go Address C:\TEMP\fr Links

The SAS System

### The GLM Procedure

**Class Level Information**

| Class | Levels | Values  |
|-------|--------|---------|
| a     | 4      | 1 2 3 4 |
| b     | 4      | 1 2 3 4 |
| c     | 4      | 1 2 3 4 |

**Number of observations** 155

---

The SAS System

### The GLM Procedure

**Dependent Variable: x**

| Source          | DF  | Sum of Squares | Mean Square | F Value | Pr > F |
|-----------------|-----|----------------|-------------|---------|--------|
| Model           | 53  | 69968.3813     | 1320.1581   | 1.41    | 0.0702 |
| Error           | 101 | 94586.5014     | 936.5000    |         |        |
| Corrected Total | 154 | 164554.8826    |             |         |        |

**R-Square Coeff Var Root MSE x Mean**

My Computer

## New SUMMARY Features

PROC SUMMARY is a workhorse procedure for many commercial data processing applications [not to mention its fans from the scientific side of the house]. Here are some of the useful features we have added.

### *Types.*

The types statement allows you to choose summary levels in between the NWAY and all-ways extremes you were limited to in previous versions. For Example:

```
PROC SUMMARY;  
  CLASS A B C D;  
  TYPES () A*( B D ) C*D D;
```

produces the following class combinations:

- O-type
- A\*B
- A\*D
- C\*D
- D.

### *Preloadfmt and Classdata*

This option will be a relief to all of you who have struggled to get the "zero rows" into your summary data set. PRELOADFMT seeds the result matrix with all values from the class variables format, while CLASSDATA seeds the result matrix from a SAS data set.

### *Autoname*

This option allows you to output many statistic for many variables, without having to type in all those variable names – AUTONAME forms output variable names by splicing the variable name and the statistic together with an "\_". For Example:

```
var a b c;  
output out=temp  
  min= max= mean=  
  / autoname;
```

will create the following output variable names: a\_min b\_min c\_min a\_max b\_max c\_max a\_mean b\_mean c\_mean.

## *Median*

PROC SUMMARY now computes the quantile statistics formerly only available in PROC UNIVARIATE.

## New TABULATE Features

New TABULATE procedure features are classified into two major categories: analytical and ODS-related.

TABULATE provides these same new analytical features that SUMMARY provides:

- Percentiles
- Ability to pre-load the class variable space using formats and/or data sets of desired class level values.
- User-defined sort orders.
- Multiple CLASS statements, with independent ordering of class variables.
- Multiple VAR statements, with independent weighting of analysis variables.

TABULATE is fully integrated with ODS. In addition to legacy listing output, output data sets and HTML are supported.

Users can create output data sets from TABULATE with either the ODS statement or the new TABULATE OUT= procedure option. The data set variables are:

- By variables from input data set.
- Class variables from input data set.
- Procedure-generated variables.
- Statistics.

With ODS HTML, TABULATE gives users a powerful means to customize the presentation of their reports, and deliver them readily to a Web-wide audience. TABULATE (and also REPORT) offers users a special feature – the ability to modify the ODS styles of their reports directly from the procedure language. The new STYLE= option permits the customization of fonts, colors, justification, etc. for various regions of a report.

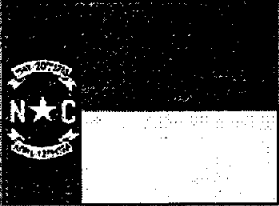
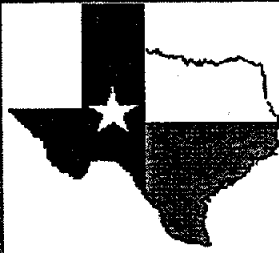
A unique feature of TABULATE is the ability to associate a user-defined format with the report data (class level values and/or statistics) to determine style attribute values. Figure xxx shows this technique being used to assign graphic images to the row

headings, and assign "traffic lights" to the table cells.

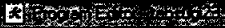
SAS

Results - C:\TEMP\b.html

### Failure Analysis by Team and Type of Failure

|                                                                                                              |                      | Median Downtime |                |         |       |
|--------------------------------------------------------------------------------------------------------------|----------------------|-----------------|----------------|---------|-------|
|                                                                                                              |                      | Oxide           | Contam-ination | Silicon | Other |
| <b>North Carolina</b><br> | <b>Overall</b>       | 4.40            | 3.29           | 0.51    | 2.39  |
|                                                                                                              | Chris Olinger        | 6.62            | 4.33           | 0.45    |       |
|                                                                                                              | David Kelley         | 1.54            | 4.71           |         | 5.36  |
|                                                                                                              | Lewis Church         | 5.08            | 0.00           |         | 0.43  |
|                                                                                                              | Martyn Wheeler       | 5.73            | 5.92           | 3.73    | 2.39  |
|                                                                                                              | Sandy McNeill        | 3.44            | 2.82           | 0.51    | 2.84  |
| <b>Texas</b><br>          | <b>Overall</b>       | 2.59            | 2.93           | 1.67    | 4.10  |
|                                                                                                              | Brian Schellenberger | 2.84            | 2.43           | 2.83    | 4.28  |
|                                                                                                              | Dan O'Connor         | 4.33            | 0.00           |         |       |
|                                                                                                              | Eric Gebhart         | 7.59            | 4.09           |         | 6.49  |
|                                                                                                              | Paul Kent            | 2.21            | 3.14           | 0.00    | 3.99  |
|                                                                                                              | Robert Ray           | 3.97            | 3.37           | 3.13    |       |






## New REPORT Features

PROC REPORT has a few new additions – one of which is for PROC REPORT in

general and the rest are for ODS (and HTML, in particular).

The general addition to PROC REPORT which can be used for both listing mode and HTML output (through ODS) is the addition of two new statements:

- compute before `_page_`
- compute after `_page_`

Have you ever wanted to get PROC REPORT to output lines above the column headers? Or ever wanted to label the top of every page with the values that start that page? Or have you ever wanted to put page footers with the values that end that page? If you've answered yes to any of the above, then you'll want one or both of the compute before/after `_page_` statements. Here's an example of "dictionary-style" page headers and footers that illustrates the user of both of these new statements.

### Page Headers and Footers

```
compute before _page_ /right;
  line dept $12.
  ,
  ,
  employee $25. ;
endcomp;

compute after _page_ /right;
  line dept $12.
  ,
  ,
  employee $25. ;
endcomp;
```

### Compute before/after page listing:

|              |               |              |
|--------------|---------------|--------------|
| Department   | Applications  | Alan Eaton   |
|              | Employee      |              |
| Applications | Alan Eaton    |              |
| IPD          | Alice         |              |
| IPD          | Catbert       |              |
| IPD          | Chris Olinger |              |
| IPD          | David Kelly   |              |
| IPD          | Dogbert       |              |
| IPD          | Don O'Connor  |              |
| IPD          | Eric Gebhart  |              |
| IPD          | Lewis Church  |              |
| IPD          |               | Lewis Church |

That was the general addition. Now let's get on to the additions for ODS which are primarily for HTML and other markup renderers.

Ever wanted to embed a graph inside PROC REPORT? How do you do that? Here's how:

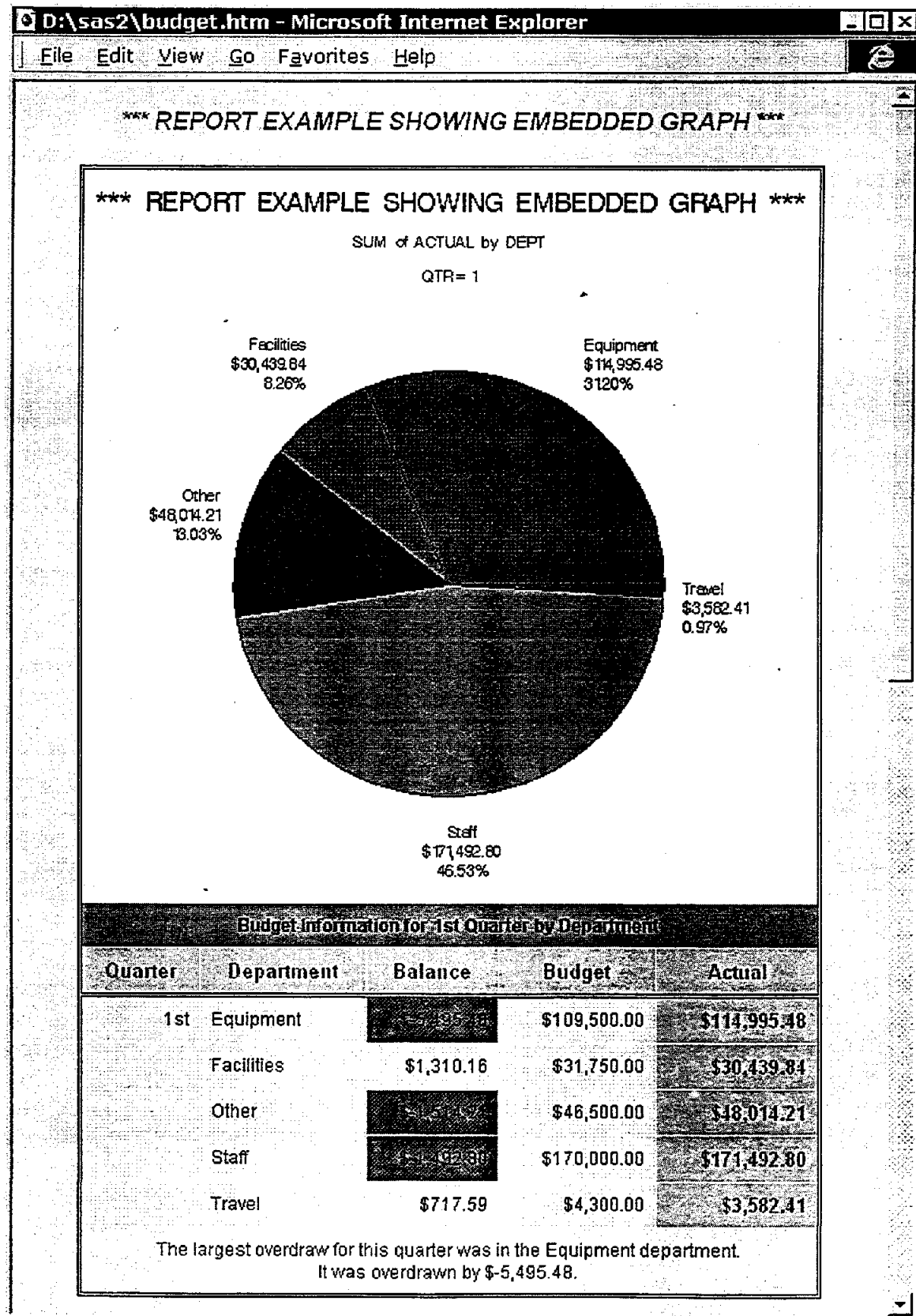
```
compute before _page_ /
  style=(background=CX0088EE
    foreground=black
    font_weight=bold
  );

if( qtr = 1 ) then
  call define( _ROW_,
    "GRSEG",
    "sasuser.budget.gchart"
  );

else if (qtr = 2 ) then
  call define( _ROW_,
    "GRSEG",
    "sasuser.budget.gchart1"
  );
```

We accomplished this by using the call define statement with the new attribute "GRSEG". Did you notice the new `_ROW_` identifier? We wanted to embed the graph in the report in place of a row, so the new `_ROW_` identifier was created for that purpose. You can also embed the graph in one of the cells of a column by using the familiar `_COL_` identifier or the name of the column.

# HTML Embedded Graph Example



Another new addition to PROC REPORT that is specific to HTML is to transform the data in a cell in an HTML Report into an HTML link. Why would you want to do that? Say you have a summary report and a separate report containing the details and you'd like to be able to link the two reports. The new "URL" attribute of the CALL DEFINE statement lets you do that.

It would look like:

```
CALL DEFINE (_COL_, "URL",
             "HTML url link string");
```

Here's an example, but keep in mind it's hard to show a link via a hard copy. The cells with the underscores are the cells with the links. The Table towards the background is a snippet of a summary table (with the patient's number data being transformed into a link to the detail table for that patient):

Now imagine "clicking" on patient number 1839. This would bring up the other table

**Patient Side Effects Log**

| patient     | event |
|-------------|-------|
| <u>183</u>  |       |
| <u>1826</u> | X     |
| <u>1839</u> | X     |
| <u>1852</u> | X     |
| <u>1865</u> | X     |
| <u>1878</u> | X     |
| <u>1891</u> | X     |
| <u>1904</u> | X     |

Side effects  
Ves  
Res

**Patient: 1839 Sex: M Drug: B Dosage: 400**

| Blood Pressure |          |           |
|----------------|----------|-----------|
| Visitdate      | Systolic | Diastolic |
| 10JAN91        | 145      | 108       |
| 17JAN91        | 147      | 108       |
| 24JAN91        | 144      | 110       |
| 31JAN91        | 141      | 108       |
| 07FEE91        | 138      | 109       |
| 14FEE91        | 135      | 105       |

My Computer

Once you get used to outputting reports in HTML, you'll be ready to snazz them up a little – maybe a report is being created for an executive and this report shows the spending of the different departments in a company. You want to draw attention to the departments that have overspent (and are "in the red"). What better way to do this than to change the background of those negative cells to red, or maybe change the font, make the text bold. All it takes is a little imagination. There's all sorts of ways to spice up the report. This is all done by the **STYLE** option.

```
STYLE <(ReportLocation)> =
  <ODS style name> {attribute*}
```

where attribute is

```
AttributeName = AttributeValue
```

Here's an example using styles to do what was mentioned above – bring attention to the negative balances in a report.

```
if balance < 0 then
  call define(_COL_,
    "style",
    "style={background=red}"
  );
```

Maybe instead of changing the style of particular cells you might want to change the style of all the columns in the report. Use the style option on the PROC REPORT statement with the COLUMN as the Report Location.

This:

```
PROC REPORT
  Data=mydata
  style(COLUMN)
    =(background=red)
  <other options>;
```

will change all the columns to having a background of red.

Or what if you want to change the style of a particular column – not all the columns? Use the style option on the define statement with COLUMN as the report location.

```
define balance /
  display
  style={background=red}
```

## SUMMARY of PROC REPORT V7

Here's a summary of the latest and greatest from PROC REPORT for version 7.

- compute before \_page\_
- compute after \_page\_
- New call define attributes:
  - "URL" -- lets you transform data into an HTML link
  - "GRSEG" -- lets you embed a graph in a row or cell of an HTML report
  - "STYLE" -- let you change the style of a row or a cell
- \_ROW\_ identifier -- used in the call define statement to embed a graph in a row or change the style of an entire row

## Your Turn

The developers, testers and documentation folk that bring you ODS are very excited about the potential of these new capabilities of the SAS System. You can send us electronic mail to [ods@unx.sas.com](mailto:ods@unx.sas.com) with your comments, or to request a copy of the example HTML files generated by a broad selection of SAS procedures, and in a variety of styles.

We hope to add more information about the ODS capabilities in future releases of SAS Software to the Research and Development section of SAS Institute's Web Page. See <http://www.sas.com/rnd>

## Trademarks

SAS, SAS/GRAPH and SAS/ACCESS are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.