Paper 159-25

Bringing Rhyme and Reason to Your Data:

A Beginner's Guide to Creating Reports that Communicate and Impress

Lora D. Delwiche, University of California, Davis

Susan J. Slaughter, Independent Consultant, Davis, CA

Introduction

In the book "The Phantom Tollbooth," a boy named Milo sees life as a waste of time. Then one day he is magically transported to the Kingdom of Wisdom. His journey starts in Expectations, then, without thinking, he finds himself in the Doldrums. During his travels, Milo meets many interesting and unusual characters, and learns about the Princess of Sweet Rhyme and the Princess of Pure Reason. The princesses have been banished from the Kingdom, but Milo rescues them thereby restoring rhyme and reason to the Kingdom. And, in so doing, Milo finds meaning in his own life.

When confronted with a large amount of data, and told to write a report, your journey will have many similarities to Milo's journey through the Kingdom of Wisdom. You, and likely your boss, will start with some expectations about what the report will be like and what information it will convey. You may end up in the Doldrums, as Milo did, not knowing how to go about producing the report. Your journey will likely take many turns, and you will learn new things along the way. But, in the end, your goal is to bring rhyme and reason to your report thereby making sense out of your data.

Expectations

One of the interesting places Milo visits during his journey is the Word Market in the city of Dictionopolis. You can buy any word you like at the Word Market, and they come in all shapes, sizes and flavors. Suppose you have been given sales data for several vendors at the market, and are expected to produce a nice report. Table 1 shows the data file in CSV (comma separated value) format. Note that the first data line contains descriptions of the data fields.

Table 1. CSV file of Word Market data.

```
Vendor,Customer,Residence,WordPurchased,WordType,DatePurchased,Quantity,Price
c,Milo,,quagmire,n,10/31/99,1,98
a,Milo,,happy,a,1/1/00,20,30
b,Whether Man,Expectations,welcome,,12/3/99,235,45
b,Duke of Definition,Dictionopolis,nonsense,n,1/8/00,26,56
b,Earl of Essence,Dictionopolis,fantastic,a,1/8/00,26,71
a,Spelling Bee,Dictionopolis,vegetable,n,3/3/00,67,9
a,King Azaz ,Dictionopolis,unabridged,a,12/24/99,1200,67
b,Humbug,Dictionopolis,balderdash,n,11/17/99,34,87
a,Dodecahedron,Digitopolis,faces,n,2/23/00,100,65
b,Mathemagician,Digitopolis,numbers,n,11/16/99,500,20
c,Wordsnatcher,Mts of Ignorance,take,v,1/16/00,,0
c,Terrible Trivium,Mts of Ignorance,work,v,1/16/00,39,22
c,Senses Taker,Mts of Ignorance,sight,n,3/10/00,450,50
b,Princess Reason,Castle in the Air,learn,v,3/19/00,1500,75
b,Princess Rhyme,Castle in the Air,wonderful,a,12/30/99,2000,50
```

Reading the Data

To start your journey, your first task is to read the data and get it into a SAS data set. The IMPORT procedure, available in Version 7 and above, makes reading CSV files easy. The IMPORT procedure is available for Windows, OS/2, UNIX and VMS operating environments. (If you are using an OS/390 system, you can read this file using the DATA step with the DSD option on the INFILE statement.) The following program reads the datafile, marketplace.csv, and creates a SAS data set named WORDSALES using the first line in the data file as variable names.

```
PROC IMPORT DATAFILE='D:\SUGI 2000\marketplace.csv'
    OUT=wordsales REPLACE;
RUN;
```

A Simple PROC PRINT

The PRINT procedure is a basic, easy to use, reporting tool available in SAS. A simple PROC PRINT, will format your data into nice columns, label the columns, and decide how to make the report best fit on the page. The basic PROC PRINT is an easy way to see if your data have been read correctly, and provides a simple report. The following, one line, program will print the WORDSALES SAS data set. The results of the PROC PRINT are shown in Table 2.

```
PROC PRINT DATA=wordsales;
RUN;
```

While this report is easier to read than the raw data, it certainly does not meet our expectations of what the finished report should look like. Here is perhaps where you find yourself in the Doldrums wondering what you can do to make this report better. When Milo found himself in the Doldrums, he discovered that the way to get out, was to start thinking. So, let's think: the Customer and Residence information is interesting, but it is not relevant to this report. The sales for each vendor are all mixed up – that's a problem. And, what's "The SAS System" doing at the top of the report? Identifying problem areas is half the battle.

Table 2. Results from simple PROC PRINT.

```
                                   The SAS System       10:56 Thursday, January 20, 2000     1


            Obs    Vendor    Customer                  Residence

              1      c       Milo
              2      a       Milo
              3      b       Whether Man               Expectations
              4      b       Duke of Definition        Dictionopolis
              5      b       Earl of Essence           Dictionopolis
              6      a       Spelling Bee              Dictionopolis
              7      a       King Azaz                 Dictionopolis
              8      b       Humbug                    Dictionopolis
              9      a       Dodecahedron              Digitopolis
             10      b       Mathemagician             Digitopolis
             11      c       Wordsnatcher              Mts of Ignorance
             12      c       Terrible Trivium          Mts of Ignorance
             13      c       Senses Taker              Mts of Ignorance
             14      b       Princess Reason           Castle in the Air
             15      b       Princess Rhyme            Castle in the Air

                   Word            Word         Date
            Obs    Purchased       Type    Purchased        Quantity          Price

              1    quagmire         n      10/31/1999              1             98
              2    happy            a      01/01/2000             20             30
              3    welcome                 12/03/1999            235             45
              4    nonsense         n      01/08/2000             26             56
              5    fantastic        a      01/08/2000             26             71
              6    vegetable        n      03/03/2000             67              9
              7    unabridged       a      12/24/1999           1200             67
              8    balderdash       n      11/17/1999             34             87
              9    faces            n      02/23/2000            100             65
             10    numbers          n      11/16/1999            500             20
             11    take             v      01/16/2000              .              0
             12    work             v      01/16/2000             39             22
             13    sight            n      03/10/2000            450             50
             14    learn            v      03/19/2000           1500             75
             15    wonderful        a      12/30/1999           2000             50
```

Sorting and Selecting Information to Print

After thinking his way out of the Doldrums, Milo soon finds himself at the Word Market. Milo discovers that you can select just about any word you want at the market, and some selections are better than others. For example, the letter X tastes like a truck full of stale air and, Milo learns, that's why people don't use X's very often. Milo also discovers how important it is to keep words sorted. After a commotion in the market place that leaves all the words in a jumble, "no say could anything one correctly" (no one could say anything correctly).

Sorting your data is one of the easiest tasks you can do, and it can also be one of the most useful for organizing your data. In our case, we want to sort by Vendor so observations for each vendor will be together. Then within Vendor, we want to sort by DatePurchased so we can see the chronology of the purchases. Here are the SAS statements to sort the WORDSALES data set:

```
PROC SORT DATA=wordsales;
  BY Vendor DatePurchased;
```

The BY statement tells SAS how to sort the observations. SAS sorts by the first variable, Vendor, then within Vendor, SAS sorts by the second variable, DatePurchased.

For this report, we decide to just look at sales to customers that live in Dictionopolis. An easy way to control which observations are used in a procedure is with the WHERE statement. For example, if you place the following statement in the PRINT procedure, SAS will only print the observations that satisfy the condition (customers that live in Dictionopolis).

```
WHERE Residence='Dictionopolis';
```

Sometimes you don't want to see all the variables in a data set when you print it. To print only selected variables, use the VAR statement in the PRINT procedure. Simply list the variables you want to print in the order that you want them to appear. For example:

```
VAR Vendor DatePurchased WordPurchased WordType
    Quantity Price;
```

Also, by default, SAS prints the observation number as a column in your PRINT procedure output. If you don't care about the observation number, it may be distracting. To drop the Obs column from your output, use the NOOBS option to the PROC PRINT statement:

```
PROC PRINT DATA=wordsales NOOBS;
```

The last thing we want to do to improve the appearance of the report, is to give the report a title. By default, SAS uses "The SAS System" as a title for all output, and SAS prints the current date at the top of each page of output. The date can be very useful, but often it is distracting. To eliminate the date, use the NODATE system option:

```
OPTIONS NODATE;
```

The OPTIONS statement can go anywhere in your program, but it is usually best to put it as the first line.

Use the TITLE statement to give your report an appropriate title. You can put the TITLE statement in the procedure, or before the PROC statement. Enclose the text you want in quotes after the TITLE keyword:

```
TITLE 'Words Purchased by Residents of
       Dictionopolis';
```

Table 3 shows the results of the following SAS program that will sort the WORDSALES data set by Vendor and DatePurchased, then print the observations where Residence equals Dictionopolis for selected variables and give the report an appropriate title:

```
OPTIONS NODATE;
PROC SORT DATA=wordsales;
  BY Vendor DatePurchased;
TITLE 'Words Purchased by Residents of
       Dictionopolis';
PROC PRINT DATA=wordsales NOOBS;
  WHERE Residence='Dictionopolis';
  VAR Vendor DatePurchased WordPurchased
      WordType Quantity Price;
RUN;
```

Table 3. Results after sorting and selecting variables and observations.

```
                           Words Purchased by Residents of Dictionopolis                        2

                            Date     Word            Word
             Vendor    Purchased     Purchased       Type         Quantity           Price

               a      12/24/1999     unabridged       a               1200              67
               a      03/03/2000     vegetable        n                 67               9
               b      11/17/1999     balderdash       n                 34              87
               b      01/08/2000     nonsense         n                 26              56
               b      01/08/2000     fantastic        a                 26              71
```

This report is much better than the last, but there is still room for improvement. It would be nice to see the actual amount the customer paid for the words they bought (Quantity * Price), and since many of those numbers are in the thousands, it would be nice to see the numbers with commas to make them easier to read. Also the actual date the word was purchased is not so important, instead you would like to see just the quarter and year.

Computing the Amount Paid

The amount the customer paid for their purchase can easily be computed in a DATA step. Use the SET statement to read the WORDSALES data set, then use an assignment statement to compute the new variable AmountPaid:

```
DATA wordsales;
  SET wordsales;
  AmountPaid=Quantity*Price;
```

Formatting Values Using Standard Formats

In the city of Dictionopolis, Milo meets King Azaz's cabinet members: the Count of Connotation, the Minister of Meaning, the Earl of Essence, the Duke of Definition, and the Undersecretary of Understanding. Whenever any of the King's advisors say a word, each one of them says the same thing using a different word. Here Milo learns that there are often many ways to represent the same idea. In the SAS language, we use formats to represent data values in different ways.

SAS has many standard formats that you can use to change the appearance of the data values in your output. For example, there are many ways to represent dates: 10/31/99, 10-31-99, 31 OCT 99, 31.10.99 to show a few. To change the format that SAS uses, you can assign a format to a variable in a FORMAT statement. You will almost always want to do this if you are using SAS date values. A SAS date value is the number of days since January 1st, 1960, and if there is no format assigned to a date variable, SAS will print just the number. You might wonder why the output we have shown so far, shows the dates in nice readable forms. The reason is that the IMPORT procedure assigns a date format (MMDDYY10.) to variables that appear to be dates. This is a nice feature of the IMPORT procedure. Since we want to print the date as the year and quarter, we will assign the variable, DatePurchased, the format YYQ5.. Also, we will assign the COMMA8. format to AmountPaid. The FORMAT statement:

```
FORMAT DatePurchased YYQ5. AmountPaid COMMA8.;
```

Table 4 shows the results of the following program, which computes a new variable, AmountPaid, and uses formats to change the appearance of the values in the report.

```
DATA wordsales;
  SET wordsales;
  AmountPaid=Quantity*Price;
PROC PRINT DATA=wordsales NOOBS;
  WHERE Residence='Dictionopolis';
  VAR Vendor DatePurchased WordPurchased
       WordType Quantity Price AmountPaid;
  FORMAT DatePurchased YYQ5. AmountPaid COMMA8.;
RUN;
```

Summarizing Data and Labeling Data Using User-Defined Formats

Our report is looking better, but it would be nice to know which Vendor is which. Vendor "a" is not very descriptive. Vendor "Spelling B Ranch" would be a lot better. In Dictionopolis, Milo meets Faintly Macabre, the Official Which. The job of the Official

Table 4. Word Market sales data using formats.

```
                          Words Purchased by Residents of Dictionopolis                         3

                 Date        Word            Word                                          Amount
       Vendor    Purchased   Purchased       Type        Quantity            Price           Paid

         a         99Q4      unabridged       a             1200                67         80,400
         a         00Q1      vegetable        n               67                 9            603
         b         99Q4      balderdash       n               34                87          2,958
         b         00Q1      nonsense         n               26                56          1,456
         b         00Q1      fantastic        a               26                71          1,846
```

Which is to decide which words to use for every occasion. When Faintly was a young Which, she had trouble determining which words to use, and would often use too few. Faintly ended up in prison for dishing out so few words that no one could say a thing. When you write your SAS program to create a report, you can play the role of the Official Which by deciding what SAS prints. But don't end up in Faintly's position by using too few words.

Both the Vendor and WordType variables have coded values, which saves space in data files, but is not very useful in reports. Here is a case where too little, just a single character, could put you in a difficult spot because no one will understand the report. To print out a better description of the data value, create a user-defined format using the FORMAT procedure, and apply the format to the variable in a FORMAT statement.

The following FORMAT procedure creates two character formats: $ven. and $wt.. When applied, the $ven. format causes SAS to print "Spelling B Ranch" instead of printing the actual variable's value, "a". Likewise, the $wt. format causes SAS to print "noun" instead of "n". Notice that in the FORMAT procedure, the format names do not end in a period, but when you use them in a FORMAT statement, the formats must end in a period.

```
PROC FORMAT;
 VALUE $ven 'a'= 'Spelling B Ranch'
            'b'= 'Sentences Farm'
            'c'= 'Typos R Us';
 VALUE $wt  'n'= 'Noun'
            'v'= 'Verb'
            'a'= 'Adjective'
            OTHER = 'Other';
RUN;
```

To apply the formats to the proper variables, use the FORMAT statement:

```
FORMAT DatePurchased YYQ5. AmountPaid COMMA8.
       Vendor $ven. WordType $wt.;
```

Another thing you can do with the PRINT procedure is summarize your data. There are several SAS procedures that will summarize data: MEANS, UNIVARIATE, TABULATE – to name a few. But the PRINT procedure is useful if, in addition to summary information, you also want to print the observations. To get a separate summary for each Vendor, use the BY statement, and to specify which variable to sum, AmountPaid, use the SUM statement:

```
PROC PRINT DATA=wordsales NOOBS;
  BY Vendor;
  SUM AmountPaid;
```

The results of the following program are shown in Table 5. This program now prints the observations for residents of Dictionopolis using the two user-defined and two standard formats, selects which variables to print, and summarizes the data by Vendor for the AmountPaid:

```
PROC PRINT DATA=wordsales NOOBS;
  WHERE Residence='Dictionopolis';
  BY vendor;
  SUM AmountPaid;
  VAR  DatePurchased Wordtype Quantity Price
       AmountPaid;
  FORMAT DatePurchased YYQ5. AmountPaid COMMA8.
         Vendor $ven. WordType $wt.;
RUN;
```

Creating HTML Output and Choosing Styles

Now we have a nice simple looking report with the information that we want, but it could use some style and color. One of many obstacles Milo faces on his way to rescue the princesses, is the Senses Taker. The Senses Taker fools Milo and his traveling companions by making them see things that aren't there, smell odors that don't exist, and hear sounds never made. Milo learns how important the senses are, and that something visually appealing can be quite attractive.

With Version 7 and above of the SAS system, in addition to text output, you can also easily create HTML reports. HTML reports are great for posting results on the Web, and for e-mailing results to colleagues who can read HTML attachments. You can also import HTML files into Word documents (as was done in this paper). The HTML files produced from SAS procedures have more style than the regular output listing, and they have color. You can create HTML files programmatically, but if you are using a Windows version of SAS, you can simply create HTML output by changing a setting in your menu items. Go to Tools/ Options/ Preferences from the pull-down menus, then select the Results tab. Put a check mark next to "Create HTML". Now HTML output will appear when you run your SAS programs. To save the HTML output to a file, make the HTML Results Viewer window active (click on it), then, click on the File menu item and choose "Save As…".

Table 6 shows the first part of the previous output using the default HTML style. If you are reading this paper in the printed version of the Proceedings, you will see a gray scale version. Only the first part of the output is shown here to save space. To change styles for the HTML files, go to the same Results tab (Tools/Options/Preferences) and choose a style from the Style list. Table 7 shows what the brick style looks like. Again, you will see a gray scale version if you are reading the printed version of the Proceedings (variable names are in brick red):

If you are not using SAS in the Windows operating environment, or if you want more control over the HTML files SAS creates, you can use ODS HTML statements in your program. To generate HTML output all you need are two statements—one to open the HTML file, and one to close it. There are other types of ODS statements, and other types of HTML files that SAS can write, not to mention, other types of output (such as PostScript, Rich Text Format, and output data sets). However, in this example we

Table 5. Results showing use of user defined formats and summarizing data using PROC PRINT.

```
                        Words Purchased by Residents of Dictionopolis                      4

------------------------------------ Vendor=Spelling B Ranch -----------------------------------

                 Date                                                       Amount
               Purchased     WordType          Quantity           Price      Paid

                  99Q4       Adjective             1200              67     80,400
                  00Q1       Noun                    67               9        603
               ---------                                                  --------
                 Vendor                                                     81,003


------------------------------------ Vendor=Sentences Farm ------------------------------------

                 Date                                                       Amount
               Purchased     WordType          Quantity           Price      Paid

                  99Q4       Noun                    34              87      2,958
                  00Q1       Noun                    26              56      1,456
                  00Q1       Adjective               26              71      1,846
               ---------                                                  --------
                 Vendor                                                      6,260
                                                                          ========
                                                                            87,263
```

simply want to create an HTML file containing the report. We do that with these two basic statements.

```
ODS HTML BODY = 'bodyfile.html';

ODS HTML CLOSE;
```

The ODS HTML statement opens a file, called a body file, which will contain the report. With the ODS HTML statement, you can create other HTML files, such as a table of contents, but in this paper we just want the report itself. The ODS HTML CLOSE statement closes the file when the report is done. Generally speaking, you want to put the first statement just before the procedure, and the closing statement just after the RUN statement, but before any other procedures. Here is a PROC PRINT with ODS HTML statements.

```
ODS HTML BODY = 'c:\MyHTML\marketplace.html';
PROC PRINT DATA=wordsales NOOBS;
  WHERE Residence='Dictionopolis';
  BY vendor;
  SUM AmountPaid;
  VAR  DatePurchased Wordtype Quantity Price
       AmountPaid;
  FORMAT DatePurchased YYQ5. AmountPaid COMMA8.
         Vendor $ven. WordType $wt.;
RUN;
ODS HTML CLOSE;
```

SAS will write the output from PROC PRINT in a file named marketplace.html in a directory named MyHTML on the C drive. The resulting HTML file is the same as the file you would get by using the HTML option in the pull down menus (shown in Table 6).

This report uses the DEFAULT style. You can specify a different style in the ODS statement using a STYLE= option:

```
ODS HTML BODY='bodyfile.html' STYLE=style-name;
```

where style-name is the name of one of the styles provided with SAS software. The following styles are currently available.

ODS Styles

DEFAULT	BROWN	STATDOC	MINIMAL
BEIGE	D3D	BRICK	

Summary Reports

When Milo visits the city of Digitopolis, he learns about the importance of numbers. The Mathemagician shows him around the numbers mine, and serves him subtraction stew (the more you eat the hungrier you get). And, while trying to reach Infinity, Milo meets half a boy, neatly divided from top to bottom. The boy explains that he is from the average family with 2.58 children, and he is the .58. Milo begins to realize that sums and averages—and all things mathematical—are important in understanding the world he lives in.

While you can produce some statistics with PROC PRINT, including sub-totals and grand totals, you can't produce a report containing only summary data without any details about individual observations. For a summary report you need to use a different procedure.

PROC MEANS (and its nearly identical twin SUMMARY), FREQ, REPORT, and TABULATE each produce summary reports. In this article, we focus on PROC TABULATE because it is relatively easy to learn, produces attractive tabular reports that are well-suited to the web, and its output can be customized using the powerful STYLE= option. In other words, PROC TABULATE is a good procedure for both rhyme and reason—rhyme to make your reports look impressive, and reason to turn data into information.

Table 6. HTML report using the DEFAULT style.

Words Purchased by Residents of Dictionopolis

Vendor=Spelling B Ranch

DatePurchased	WordType	Quantity	Price	AmountPaid
99Q4	Adjective	1200	67	80,400
00Q1	Noun	67	9	603
Vendor				**81,003**

Table 7. HTML report using the BRICK style.

Words Purchased by Residents of Dictionopolis

Vendor=Spelling B Ranch

DatePurchased	WordType	Quantity	Price	AmountPaid
99Q4	Adjective	1200	67	80,400
00Q1	Noun	67	9	603
Vendor				**81,003**

A Simple PROC TABULATE

Here is a simple PROC TABULATE using the WORDSALES data. Notice that this program contains a FORMAT, and creates an HTML file using the MINIMAL style.

```
ODS HTML BODY = 'c:\MyHTML\marketplace1.html'
    STYLE=MINIMAL;
* Simple 2-way PROC TABULATE;
TITLE ’Word Market Sales’;
PROC TABULATE DATA = wordsales MISSING;
   CLASS WordType Vendor;
   TABLE WordType, Vendor;
   FORMAT WordType $wt. Vendor $ven.;
RUN;
ODS HTML CLOSE;
```

The MISSING option in the PROC statement tells SAS to include missing values in the report. Without the MISSING option, SAS will, by default, exclude observations with missing values for any CLASS variable. The CLASS statement lists the categorical variables (often character variables) for SAS to use when dividing observations into groups.

The TABLE statement specifies how the report should be organized, and what numbers to compute. You can have more than one TABLE statement, and each TABLE statement will create a separate report. Each TABLE statement can specify up to three dimensions—a page dimension, a row dimension, and a column dimension, in that order—and these dimensions are separated by commas. If you specify only one dimension, SAS will use that for columns. With two dimensions, you get rows and columns. In this TABLE statement, there are only two dimensions, so the report will contain one row for each value of WordType, and one column for each value of Vendor. If there were a third dimension, the report would contain one page for each value of that variable. The output appears in Table 8.

A word of advice, you should always build your PROC TABULATE one feature at a time. That means you start with the column variable. When that is running correctly, then add the row variable. When the column and row dimensions are correct, then add the page variable, if you need one. Then you can add statistics and style options—one at a time. Note that to avoid scrambling your table, you must insert the page and row variables IN FRONT of the column variable in your TABLE statement.

A Compound PROC TABULATE

The previous PROC TABULATE contained only categorical variables. To add continuous variables (which must be numeric so you can compute sums and means), list them in a VAR statement. You can have both a CLASS and a VAR statement, or just one, but all variables listed in a TABLE statement must appear in either a CLASS or a VAR statement.

Table 8. The output from a simple PROC TABULATE.

Word Market Sales

	Vendor		
	Spelling B Ranch	Sentences Farm	Typos R Us
	N	N	N
WordType			
Other	.	1	.
Adjective	2	2	.
Noun	2	3	2
Verb	.	1	2

By default, SAS will compute the number of observations in a group for CLASS variables (as it did in the preceding example), or the sum of the values for a variable listed in a VAR statement, but you can choose many other statistics. These are a few of the values that PROC TABULATE can compute and the keywords to request them.

Keyword	Value
ALL	adds a row, column, or page showing the total
MEAN	the arithmetic mean
MEDIAN	the median
N	the number of non-missing values
P90	the 90th percentile
PCTN	the percentage of observations for that group
PCTSUM	the percentage of a total sum for that group
SUM	the sum

You can insert these keywords into a TABLE statement right along with the variables. Also, variables and keywords can be crossed, concatenated, and grouped. To concatenate variables, simply separate them with a space. For example:

```
TABLE WordType ALL;
```

To cross variables, separate them with an asterisk. For example:

```
TABLE MEAN * WordType;
```

To group variables, enclose them in parentheses. This TABLE statement contains concatenating, crossing, and grouping.

```
TABLE MEAN * (WordType Vendor);
```

The following PROC TABULATE uses these features, and the results appear in Table 9.

```
ODS HTML BODY = 'c:\MyHTML\marketplace2.html'
    STYLE=MINIMAL;
* Compound PROC TABULATE;
PROC TABULATE DATA = wordsales MISSING;
   CLASS WordType Vendor;
   VAR Quantity AmountPaid;
   FORMAT WordType $wt. Vendor $ven.;
   TABLE WordType ALL,
      Vendor * SUM*(Quantity AmountPaid);
RUN;
ODS HTML CLOSE;
```

This TABLE statement contains only two dimensions: rows and columns. In the row dimension, the variable WordType is separated from the keyword ALL by a space so SAS will concatenate these values, printing the summary line for ALL right below the values of WordType. In the column dimension, the variable, Vendor, is crossed with the keyword SUM, which is crossed with the variables Quantity and AmountPaid. Quantity and AmountPaid are grouped together by parentheses, and appear side by side in the table.

Formatting Data Cells and Changing Headers

The report shown in Table 9 has some good data in it, but it doesn't look very pretty. Now it's time to add a little rhyme to the reason. The following program produces the report shown in Table 10.

```
ODS HTML BODY = 'c:\MyHTML\marketplace3.html'
    STYLE=MINIMAL;
PROC TABULATE DATA = wordsales MISSING
     FORMAT=10.0;
   CLASS WordType Vendor;
   VAR Quantity AmountPaid;
   FORMAT WordType $wt. Vendor $ven.;
   TABLE WordType ALL,
      Vendor='Totals by Vendor' * SUM=' '
      *(Quantity AmountPaid*FORMAT=DOLLAR12.2)
      / MISSTEXT = 'no sales';
RUN;
ODS HTML CLOSE;
```

The FORMAT statement we have been using specifies a format for the column and row headers, in this case, the user-defined formats created earlier. In contrast, the FORMAT= option in the PROC statement specifies a format for all data cell values, the standard format 10.0. Yet another FORMAT= option appears in the TABLE statement specifying the standard format, DOLLAR12.2, for the variable AmountPaid. The format option in the TABLE statement will override the format in the PROC statement for AmountPaid, but SAS will still use the 10.0 format for the variable Quantity.

Table 9. The output from a compound PROC TABULATE.

Word Market Sales

	Vendor					
	Spelling B Ranch		Sentences Farm		Typos R Us	
	Sum		Sum		Sum	
	Quantity	AmountPaid	Quantity	AmountPaid	Quantity	AmountPaid
WordType						
Other	.	.	235.00	10575.00	.	.
Adjective	1220.00	81000.00	2026.00	101846.00	.	.
Noun	167.00	7103.00	560.00	14414.00	451.00	22598.00
Verb	.	.	1500.00	112500.00	39.00	858.00
All	1387.00	88103.00	4321.00	239335.00	490.00	23456.00

The column headers in the report shown in Table 9 are rather confusing. You can customize them in the TABLE statement. This program changes the header for Vendor to 'Totals by Vendor' and then sets the header for SUM to blank. SAS will just delete the row labeled Sum, making the report look cleaner.

The MISSTEXT= option in the TABLE statement tells SAS to print the words "no sales" in any cells with missing values. Without the MISSTEXT= option, SAS prints a period for missing numeric data.

Table 10. Output from PROC TABULATE with formatted data cells and custom headers.

Word Market Sales

	Totals by Vendor					
	Spelling B Ranch		Sentences Farm		Typos R Us	
	Quantity	AmountPaid	Quantity	AmountPaid	Quantity	AmountPaid
WordType						
Other	no sales	no sales	235	$10,575.00	no sales	no sales
Adjective	1220	$81,000.00	2026	$101,846.00	no sales	no sales
Noun	167	$7,103.00	560	$14,414.00	451	$22,598.00
Verb	no sales	no sales	1500	$112,500.00	39	$858.00
All	1387	$88,103.00	4321	$239,335.00	490	$23,456.00

Customizing HTML Output with STYLE= options

When Milo visits the forest of sight, he meets Chroma the Great and learns the importance of colors. Chroma conducts the orchestra that “plays” the colors of the kingdom. Each morning, Chroma conducts the sunrise and the orchestra plays all day through the sunset. When Milo takes a turn conducting, he learns how terribly things can go wrong: flowers turn black, rocks are chartreuse, even his trusty travel companion, Tock the Watch Dog, turns to a magnificent ultra-marine. Colors are important, but the wrong color in the wrong place can send the wrong message.

Up to this point, all the options we’ve used in PROC TABULATE can be used with standard SAS listing output, but there’s a limit to what you can do with standard SAS listing output. You can’t, for example, use color. To do that you need a different type of output such as HTML, and special options that are part of ODS.

The STYLE= option is one way to customize HTML output. When used in an ODS statement, the STYLE= option controls the overall look of a report. You can also use the STYLE= option in PROC TABULATE (and in PROC REPORT, but not in other procedures) to control specific features of your reports. The following program uses the STYLE= option in the PROC and CLASS statements, and produces the report in Table 11.

```
ODS HTML BODY = 'c:\MyHTML\marketplace4.html'
    STYLE=MINIMAL;
* Add STYLE= options;
PROC TABULATE DATA = wordsales MISSING
   FORMAT = 10.0
   STYLE={FOREGROUND=RED FONT_WEIGHT=BOLD
      FONT_FACE=COURIER};
   CLASS WordType Vendor /
      STYLE={FONT_STYLE=ITALIC};
   VAR Quantity AmountPaid;
   FORMAT WordType $wt. Vendor $ven.;
   TABLE WordType ALL,
      Vendor='Totals by Vendor' * SUM=' '
      *(Quantity AmountPaid*FORMAT=DOLLAR12.2)
      / MISSTEXT = 'no sales';
RUN;
ODS HTML CLOSE;
```

When in a PROC statement, STYLE= affects the data cell values; when it appears in a CLASS statement, the STYLE= option affects the headers for class variables. Using other STYLE= options, you can customize practically any feature in PROC TABULATE output. Here are a few examples of the features that you can control with the STYLE= option in PROC TABULATE.

STYLE= option	Example of value
BACKGROUND	BLACK
FOREGROUND	ORANGE
FONT_FACE	COURIER
FONT_WEIGHT	BOLD
FONT_STYLE	ITALIC
JUST	C
VJUST	C

Table 11. Output from PROC TABULATE with STYLE= options.

Word Market Sales

	Totals by Vendor					
	Spelling B Ranch		Sentences Farm		Typos R Us	
	Quantity	AmountPaid	Quantity	AmountPaid	Quantity	AmountPaid
WordType						
Other	no sales	no sales	**235**	**$10,575.00**	no sales	no sales
Adjective	**1220**	**$81,000.00**	**2026**	**$101,846.00**	no sales	no sales
Noun	**167**	**$7,103.00**	**560**	**$14,414.00**	**451**	**$22,598.00**
Verb	no sales	no sales	**1500**	**$112,500.00**	**39**	**$858.00**
All	**1387**	**$88,103.00**	**4321**	**$239,335.00**	**490**	**$23,456.00**

Adding traffic-lighting with a STYLE= option

One of the most impressive effects you can create with the STYLE= option is traffic-lighting. With traffic-lighting, the values of the data determine their appearance. In this way, you can draw attention to the most important data, or highlight relationships between the values. To add traffic-lighting, you create a user-defined format specifying different style attributes for each range of values, and then use that format as the value of a style element.

The following program is exactly like the preceding one except that the STYLE= option has been removed from the PROC statement. Instead, the STYLE= options appear in the TABLE statement. That way the style of Quantity and AmountPaid can be set individually. Two user-defined formats, QUANT. and PAID., are created by a PROC FORMAT. Then those formats are used as the values for FOREGROUND= in the STYLE= options in the TABLE statement. The report produced by this PROC TABULATE appears in Table 12.

```
* Create formats for traffic-lighting;
PROC FORMAT;
   VALUE quant 1000-<2000 = 'LIGHT BLUE'
               2000-high = 'RED'
               OTHER = 'BLACK';
   VALUE paid  100000-<200000 = 'LIGHT BLUE'
               200000-high = 'RED'
               OTHER = 'BLACK';
RUN;

ODS HTML BODY = 'c:\MyHTML\marketplace5.html'
    STYLE=MINIMAL;
* Add traffic-lighting;
PROC TABULATE DATA = wordsales MISSING
      FORMAT = 10.0;
   CLASS WordType Vendor /
      STYLE={FONT_STYLE=ITALIC};
   VAR Quantity AmountPaid;
   FORMAT WordType $wt. Vendor $ven.;
   TABLE WordType ALL,
      Vendor='Totals by Vendor' * SUM=' '
      *(Quantity*{STYLE={FOREGROUND=quant.
         FONT_WEIGHT=BOLD FONT_FACE=COURIER}}
      AmountPaid*{STYLE={FOREGROUND=paid.
         FONT_WEIGHT=BOLD FONT_FACE=COURIER}}
      *FORMAT=DOLLAR12.2)
      / MISSTEXT = 'no sales';
RUN;
ODS HTML CLOSE;
```

Table 12. Output from PROC TABULATE with traffic-lighting.

Word Market Sales

	Totals by Vendor					
	Spelling B Ranch		Sentences Farm		Typos R Us	
	Quantity	AmountPaid	Quantity	AmountPaid	Quantity	AmountPaid
WordType						
Other	no sales	no sales	235	$10,575.00	no sales	no sales
Adjective	1220	$81,000.00	2026	$101,846.00	no sales	no sales
Noun	167	$7,103.00	560	$14,414.00	451	$22,598.00
Verb	no sales	no sales	1500	$112,500.00	39	$858.00
All	1387	$88,103.00	4321	$239,335.00	490	$23,456.00

Conclusions

After rescuing Rhyme and Reason, Milo returns home. Inspired by his adventures, Milo finds that his life now has meaning. We hope that you feel inspired to find rhyme and reason in your data. By sorting, selecting, and summarizing; and by controlling style, font, and color, you can reduce a jumble of data into concise and meaningful reports that go beyond simple text to become information visualization.

References

Delwiche, Lora D. and Susan J. Slaughter (1998). *The Little SAS Book: A Primer, Second Edition*. SAS Institute, Cary, NC.

Kelley, David and Sandy McNeill (1999). Getting Stylish with Version 7 Base Reporting. *Proceedings of the Twenty-Fourth Annual SAS Users Group International Conference*. SAS Institute, Cary, NC.

Norton, Juster (1961). *The Phantom Tollbooth*. Random House, NY.

About the Authors

Lora Delwiche and Susan Slaughter are authors of *The Little SAS Book: A Primer* published by SAS Institute. They may be contacted at:

Lora D. Delwiche	(530) 752-6285 llddelwiche@ucdavis.edu
Susan J. Slaughter	(530) 756-8434 sjslaughter@mother.com