Using Data Set Labels to Make Your SAS® Output Self Documenting
Stanley E. Legum, Ph.D., Westat, Rockville, MD

ABSTRACT
While programmers are frequently encouraged to use comments and other techniques to make their code self-documenting, they often neglect to document their data sets and output. SAS® data set labels are an easy-to-use but often overlooked documentation technique. This paper describes how to create data set labels and how to use them in title and footnote statements to document report output. The technique employs PROC SQL and macro functions.

INTRODUCTION
Programmers are, with good reason, encouraged to make their code self-documenting through the liberal use of comments and other means. They learn to use comments in their code and to assign meaningful variable names and descriptive variable labels. Two areas that are often overlooked for self-documentation are output and data sets. This paper discusses how documenting data sets using internal data set labels can help to document both the code and the output, as well as providing the immediate benefit of documenting the data sets.

Output can be made self-documenting through the use of both titles and footnotes. What goes into each is a matter of project and programmer style, but the result should be that the user should have no doubt what the data in the output represent. Ideally titles and footnotes will also provide a programmer with enough information to reproduce the output, if required. This means being able to identify the input data sets and the source code.

Data sets can be made at least partially self-documenting by selecting meaningful data set names, by assigning SAS internal data set labels, and, as in code, assigning meaningful variable names and descriptive variable labels.

Output can be documented through titles and footnotes. These should, of course, be used to descriptively label the output so that the target audience knows what it is looking at. To some extent, what to include in the titles and footnotes is fairly clear. If the output is an age distribution subset of men who enrolled in a clinical trial in 2007, then you will probably have a high-level title naming the study (“XYZ CLINICAL TRIAL”), a second label indicating that the data represent 2007 data, and a third label indicating that the table is limited to data for men. If you use PROC FREQ to generate the distribution, the procedure will automatically display the variable name and, if it exists, the variable label. You may also use a format to group the ages into clearly named ranges. In the context of an ongoing study, this may be sufficient to unambiguously communicate to the recipient of the output all that he or she needs to know to identify it.

But what happens when someone comes to you six months later with a copy of your report and asks you to prepare a parallel report for women or to generate a comparable report on an updated version of the data set? A useful technique that can avoid the need to search through complex directory structures or old paper files is to include the path and data set name of the source code in a footnote (or a title), if the report format constraints permit it. This will at least let you quickly find the source code in order to review it, make any needed modifications, and then produce the needed output.

Where multiple versions of the same or similar data sets are involved, it is also useful to identify the input data sets used to produce the report either in a title or a footnote. You can, of course, write the appropriate descriptive title or footnote each time you rerun the program. Alternatively, with a relatively small amount of work, you can create a macro variable that is used both to call the data set and to identify the data set in a title or footnote. If your data set names are sufficiently descriptive, this may be all you need to do. On the other hand, if you create descriptive internal data set labels, you can use these labels in your output documentation.
USING INTERNAL DATA SET LABELS TO DOCUMENT CODE AND DATA SETS

The SAS® System provides an excellent means for documenting data sets in the internal labels for SAS data sets. Internal data set labels are easy to create using the LABEL = data set option. They can be viewed using the CONTENTS and DATASETS procedures. They can also be read from dictionary tables during a SAS job using PROC SQL. This makes them available for use in title and footnote statements for documenting output.

To create a data set label, simply specify a label option in the data statement. For example, the statement

```sas
data FileEX1 (label = "This file was created to demonstrate the ease of creating data set labels");
```

creates a data set label that reads, “This file was created to demonstrate the ease of creating data set labels.” Since most of us see data set labels only near the top of a PROC CONTENTS listing, these labels might appear to have only limited use. There are, however, a number of reasons why SAS programmers should get into the habit of labeling all data sets:

- Data set labels serve as comments in the code.
- Data set labels provide a descriptive clue to the reason why a data set created only for testing purposes exists, when it was accidentally not deleted and is discovered at some time in the future.
- For permanent data sets, the data set label provides a means of more fully describing the contents of a data set than can be done with the data set name alone. For example, in a longitudinal study, a label can provide useful information about dates and release numbers as in:
  - XYZ Study Demographic Data thru Dec 2005 - Release XYZREL0506.02
  - XYZ Study Cancer Registry Diagnoses thru December 31, 2002 from Registries 2005. Release XYZREL0512.02
- Data set labels provide a means of more fully documenting your output.

DOCUMENTING OUTPUT

There are, of course, numerous ways to document output. These include the familiar display of variable names and variable labels, the display of raw or formatted variable values, the identification of the SAS procedure that produced the output, and the date on which the output is created. While not all of these things appear as the output for every SAS procedure and many which appear by default can be overridden, they occur often enough to be all too easily taken for granted.

Programmers quickly learn to use at least some titles when writing their programs and at least some supplement titles with footnotes. There are two main reasons for creating clear titles and footnotes:

- They identify the report or data source and/or provide explanations useful to the customer.
- They help the programmer remember how the report was generated when he or she is shown a copy at a later date.

Titles and footnotes can be used to:

- Identify the project.
- Identify documents (or records in a project metadata database) containing program specifications.
- Identify the purpose of the run.
Identify the data source.

Document any subsetting performed for the analysis.

Document the meaning of codes or abbreviations.

Document the program name.

Document the program location.

Document the name or location of the document containing the detailed programming specifications or the request for the analysis.

Exactly what is appropriate in any given situation will, of course, depend on the corporate and programming environment and the purpose of the program. In general, the following guidelines have proved useful in many situations:

Title 1: Study name
Title 2: Specification ID number
Title 3: General description of purpose of run
Title 4: Description of specific table/list

Footnote 1: Input file #1
Footnote 2: Input file #2:
Footnote 3: Input file #3
...
Last footnote: Program name with full directory path

If a detailed specification or programming request exists and can be easily identified, referencing the study title and the specification ID may be sufficient.

REAL WORLD EXAMPLE
As a real world example, Westat is currently supporting a multi-phase longitudinal study involving tens of thousands of respondents who have been interviewed two or three times each with several years between each set of interviews. The work is being supported by multiple organizations and many researchers from diverse backgrounds are using the data. The following are some of the features of this study that motivate us to put a substantial amount of information in the data set labels:

- We have multiple releases of the same data sets. More precisely, we have multiple releases of data sets representing the same variables for the same respondents. This occurs for a number of reasons:
  - We receive new data from state vital statistics registries and cancer registries;
  - We create or revise derived variables; and
  - Ongoing editing.
- There are up to 15 files in each release.
- There is a project policy that all published papers identify the release number of the data set.

To minimize the need for revising code, each redelivered data set has the same file name as the previous version of the same data set. To clearly identify which version is being used, the project uses internal data set labels that identify the following:

- The study;
- The study phase (e.g., 1, 2, 3);
- The content, (e.g., responses to questionnaire; mortality data); and
- The release number.
CODE TO USE DATA SET LABELS IN FOOTNOTES
The code fragment in Figure 1 generates the table in Figure 2.

```
title1 "XYZ HEALTH STUDY" ;
title2 "DISTRIBUTION OF RESPONDENT TYPES BY SITE" ;
footnote1 "Using Demographic Data thru 2005 - Release REL0506.02"
footnote2 "\\RK25\\VOL2501\\SASModels\\FootnoteGen.sas" ;

proc freq data = P2Demo.Demographic ;
title2 "DISTRIBUTION OF RESPONDENT TYPES BY SITE" ;
tables site * RespType ;
run ;
```

Figure 1. Sample code fragment

```
XYZ HEALTH STUDY
DISTRIBUTION OF RESPONDENT TYPES BY SITE

<table>
<thead>
<tr>
<th>SITE(Site)</th>
<th>RespType(Type of Participant)</th>
<th>Frequency</th>
<th>C</th>
<th>P</th>
<th>S</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td>4916</td>
<td>31877</td>
<td>21771</td>
<td>58564</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
<td>0</td>
<td>20518</td>
<td>10576</td>
<td>31094</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>4916</td>
<td>52395</td>
<td>32347</td>
<td>89658</td>
<td></td>
</tr>
</tbody>
</table>

Using Demographic Data thru 2005 - Release REL0506.02
\\RK25\\VOL2501\\SASModels\\FootnoteGen.sas
```

Figure 2. Output produced by the code in Figure 1.

Actually, rather than write the code for footnote 1 as shown in the code fragment, we read the data set label from the Demographic File into a macro variable and then use that variable in the footnote statement. Our first approximation to writing this footnote statement looked like:

```
footnote1 "Using &FN1_LBL" ;
```

However, to account for the fact that some of our internal data set labels included commas (which wreaked havoc when interpreting the macro variable) we revised the statement to read:

```
footnote1 "Using %QSYSFUNC(trim(%SUPERQ(FN1_LBL)))" ;
```
This %SUPERQ function ensures that special characters in the label appear in the label and are not interpreted by the macro processor. (Note that, even though %SUPERQ works on macro variables, no ampersand is placed before the variable FN1_LBL.) The TRIM function was needed because the way in which we created the macro variable made it longer than the actual text. The %QSYSFUNC function allows us to use the data step function TRIM in the footnote statement although it is not in a data step.

The key for placing the contents of the data set label into a macro variable is to use SAS SQL to pull the relevant information from the SAS dictionary tables. The key statement in this process is:

```
select memlabel into :FN1_LBL
```

which creates the macro variable &FN1_LBL.

The following example demonstrates how to use PROC SQL to pull the internal data set labels from one file (Demographic.sas7bdat) in a directory assigned the libname P2Demo and two files (Mortality.sas7bdat and CancerRegistry.sas7bdat) stored in a directory assigned to libname "p2gen". Because some versions of SAS stored all the values in the dictionary tables in uppercase, the UPCASE function ensures that the values stored in the LIBNAME and MEMNAME fields in the dictionary tables will match the target values.

```
proc sql noprint;
  select memlabel into :FN1_LBL
  from dictionary.tables
  where upcase(libname) = upcase("p2demo") and
       upcase(memname) = upcase("Demographic");

  select memlabel into :FN2_LBL
  from dictionary.tables
  where upcase(libname) = upcase("p2gen") and
       upcase(memname) = upcase("Mortality");

  select memlabel into :FN3_LBL
  from dictionary.tables
  where upcase(libname) = upcase("p2gen") and
       upcase(memname) = upcase("CancerRegistry");
quit;
```

Assuming that you want to document the use of all three of these data sets in your code, as well as the name and the location of the source code, you could use the following set of footnotes:

```
footnote1 "Using %QSYSFUNC(trim(%SUPERQ(FN1_LBL)))";
footnote2 "Using %QSYSFUNC(trim(%SUPERQ(FN2_LBL)))";
footnote3 "Using %QSYSFUNC(trim(%SUPERQ(FN3_LBL)))";
footnote4 "\RK25\VOL2501\SASModels\FootnoteGen.sas";
```
The program location and file name can also be automatically generated using PROC SQL and the dictionary.extfiles table. The following code puts this information in the macro variable &FLNM.

```sql
select xpath into :FLNM
from dictionary.extfiles
where index(upcase(xpath), ".SAS") > 0 ;
```

CONCLUSIONS AND RECOMMENDATIONS
Data set labels are very easy to create; and using them has many benefits for the programmer. When using code such as that shown above, data set labels are also relatively easy to use in output. Doing so can benefit both the programmer and the consumers of the output.

Programmers will find that their work is more professional and easier to maintain if they adopt the following recommendations:

- Document files and output as well as code.
- Always attach an internal label to permanent SAS data sets.
- Provide useful output documentation including:
  - Project name;
  - Report descriptions;
  - Name and location of source code; and
  - Data sources.

DISCLAIMER
The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

ACKNOWLEDGMENTS
SAS and all other SAS Institute, Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies. Thanks to Mike Rhoads and Ian Whitlock for their review and suggestions.

CONTACT INFORMATION
Please contact the author if you have any questions or comments:
Stanley Legum
Westat
1600 Research Blvd.
Rockville, MD 20850
selegum@Westat.com