

# SAS Programming (BIOL-4V190)

## Chapter 14 Displaying Your Data

## 14.1 Introduction

So far, we have used PROC PRINT to create basic printouts of the data.

By using other statements and options that are available in the procedure, we can create nicer and more informative data displays.

## 14.2 The Basics

The most basic form of the print procedure is:

```
PROC PRINT;  
RUN;
```

This prints all variables and all observations in the data set.

Note: The data set LEARN.SALES has 15 observations.

Printouts of the data in the book only show 14 observations.

### 14.3 Changing the Appearance of Your Listing

Use the VAR statement to select or limit the variables to be included in the output and to control the print order of the variables.

By default, an Obs column which displays the Observation number is displayed in PROC PRINT output.

One way to eliminate the Obs column is to add an ID statement.

A variable listed on the ID statement should not be also listed on the VAR statement, otherwise it will appear twice in the data display.

Syntax:

```
proc print;  
  id variablename1 variablename2...variablename'n';  
  var variablename1 variablename2...variablename'n';  
run;
```

Program 14-3 provides an illustration of this and the output is shown on page 265.

Another way to eliminate the Obs column is to use the noobs option on the PROC PRINT statement.

Syntax:

```
proc print noobs;  
  var variablename1 variablename2...variablename'n';  
run;
```

```

* Omitting the Obs column using NOOBS option;
proc print data=learn.sales noobs;
  var EmpID Customer TotalSales;
run;

```

The data listing is identical to that produced by Program 14-3.

The screenshot displays the SAS Enterprise Guide interface. The 'Project Explorer' on the left shows a project structure with 'chapter 14' containing 'Log', 'Listing - chapter 14', 'HTML - chapter 14', 'PDF - chapter 14', and 'RTF - chapter 14'. The main window shows the 'Listing - chapter 14' output, titled 'Listing of SALES using the NOOBS option'. The listing is a table with three columns: 'Emp ID', 'Customer', and 'Total Sales'. The data is sorted by 'Emp ID' in ascending order. The 'Task Status' window at the bottom is empty. The status bar at the bottom indicates 'Ready' and 'justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation'.

Emp ID	Customer	Total Sales
1843	Barco Corporation	449.5
1843	Cost Cutter's	599.0
1843	Minimart Inc.	15597.0
1843	Barco Corporation	5129.0
1843	Ely Corp.	299.5
0177	Food Unlimited	699.0
0177	Shop and Drop	899.0
1843	Cost Cutter's	9109.0
9888	Cost Cutter's	299.5
9888	Pet's are Us	1990.0
0017	Roger's Spirits	19995.0
0017	Spirited Spirits	1995.0
0177	Minimart Inc.	52.5
0177	Barco Corporation	20000.0
1843	Minimart Inc.	15597.0

## 14.4 Changing the Appearance of Values

Data can be reformatted by adding a format statement in a procedure.

The format is TEMPORARILY associated with the variable for that procedure only.

If a variable already has a format PERMANENTLY associated with it, a new TEMPORARY format can be applied to the variable by using a format statement with that variable in the procedure.

To “unformat” a variable with a PERMANENT format and display the raw data value, include the variable name on a format statement without specifying a format.

Syntax:

```
proc print;  
  var variablename1 variablename2...variablename'n';  
  format variablename1 variablename4 variablename5 format. variablename2 $format.;  
run;
```

This is illustrated in Program 14-4 on page 266.

To “unformat” a variable with a PERMANENT format and display the raw data value, include the variable name on a format statement without specifying a format.

Syntax:

```
proc print;  
  var variablename1 variablename2...variablename'n';  
  format variablename1 variablename2;  
run;
```

The data set LEARN.BICYCLES has two variable that are formatted, TotalSales and UnitCost, as shown in the PROC CONTENTS output.

The screenshot displays the SAS Enterprise Guide interface. The 'Project Explorer' on the left shows the 'BICYCLES' data set. The main window shows the 'RTF - chapter 14' output, which contains the following table:

Obs in First Data Page	18
Number of Data Set Repairs	0
File Name	/courses/u_ucsd.edu/i_536036/c_629/saslib/bicycles.sas7bdat
Release Created	9.0101M3
Host Created	SunOS
Inode Number	2143051
Access Permission	rwxr-X---
Owner Name	justina.flavin
File Size (bytes)	16384

Below this table is an 'Alphabetic List of Variables and Attributes' table:

#	Variable	Type	Len	Format	Label
1	Country	Char	25		
3	Manuf	Char	10		Manufacturer
2	Model	Char	14		
6	TotalSales	Num	8	DOLLAR10.	Sales in Thousands
5	UnitCost	Num	8	DOLLAR10.	
4	Units	Num	8		

The 'Task Status' window at the bottom shows a table with columns: Task, Status, Queue, and Server.

By default, the formatted data values are printed.

**SAS Enterprise Guide**

File Edit View Code Data Describe Graph Analyze Add-In OLAP Tools Window Help

BICYCLES (Process Flow)

Project Explorer

- Project
  - Process Flow
    - chapter 14
      - Log
      - Listing - chapter 14
      - HTML - chapter 14
      - PDF - chapter 14
      - RTF - chapter 14
      - BICYCLES

Project Designer

chapter 14\* Log (chapter 14 (Process Flow)) RTF - chapter 14 Listing - chapter 14

Listing of BICYCLES 18:39 Wednesday, A

Obs	Country	Model	Manuf	Units	UnitCost	TotalSales
1	USA	Road Bike	Trek	5000	\$2,200	\$11,000
2	USA	Road Bike	Cannondale	2000	\$2,100	\$4,200
3	USA	Mountain Bike	Trek	6000	\$1,200	\$7,200
4	USA	Mountain Bike	Cannondale	4000	\$2,700	\$10,800
5	USA	Hybrid	Trek	4500	\$650	\$2,925
6	France	Road Bike	Trek	3400	\$2,500	\$8,500
7	France	Road Bike	Cannondale	900	\$3,700	\$3,330
8	France	Mountain Bike	Trek	5600	\$1,300	\$7,280
9	France	Mountain Bike	Cannondale	800	\$1,899	\$1,519
10	France	Hybrid	Trek	1100	\$540	\$594
11	United Kingdom	Road Bike	Trek	2444	\$2,100	\$5,132
12	United Kingdom	Road Bike	Cannondale	1200	\$2,123	\$2,548
13	United Kingdom	Hybrid	Trek	800	\$490	\$392
14	United Kingdom	Hybrid	Cannondale	500	\$880	\$440
15	United Kingdom	Mountain Bike	Trek	1211	\$1,121	\$1,358
16	Italy	Hybrid	Trek	700	\$690	\$483
17	Italy	Road Bike	Trek	4500	\$2,890	\$13,005
18	Italy	Mountain Bike	Trek	3400	\$1,877	\$6,382

Task Status

Task	Status	Queue	Server
------	--------	-------	--------

Ready justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation Line 1, Col 1

By using a null format statement, the actual unformatted data values can be displayed.

```
proc print data=learn.bicycles;
  title "Listing of BICYCLES - no formats on totalsales and unitcost";
  format TotalSales unitcost;
run;
```

**Listing of BICYCLES - no formats on totalsales and unitcost**

Obs	Country	Model	Manuf	Units	Unit Cost	Total Sales
1	USA	Road Bike	Trek	5000	2200	11000.00
2	USA	Road Bike	Cannondale	2000	2100	4200.00
3	USA	Mountain Bike	Trek	6000	1200	7200.00
4	USA	Mountain Bike	Cannondale	4000	2700	10800.00
5	USA	Hybrid	Trek	4500	650	2925.00
6	France	Road Bike	Trek	3400	2500	8500.00
7	France	Road Bike	Cannondale	900	3700	3330.00
8	France	Mountain Bike	Trek	5600	1300	7280.00
9	France	Mountain Bike	Cannondale	800	1899	1519.20
10	France	Hybrid	Trek	1100	540	594.00
11	United Kingdom	Road Bike	Trek	2444	2100	5132.40
12	United Kingdom	Road Bike	Cannondale	1200	2123	2547.60
13	United Kingdom	Hybrid	Trek	800	490	392.00
14	United Kingdom	Hybrid	Cannondale	500	880	440.00
15	United Kingdom	Mountain Bike	Trek	1211	1121	1357.53
16	Italy	Hybrid	Trek	700	690	483.00
17	Italy	Road Bike	Trek	4500	2890	13005.00
18	Italy	Mountain Bike	Trek	3400	1877	6381.80



## 14.5 Controlling the Observations That Appear in Your Listing

Use a WHERE statement to limit the data that are included in the display.

Syntax:

```
proc print;  
  where condition...;  
run;
```

Programs 14-5 and 14-6 illustrate this concept.

## 14.6 Adding Additional Titles and Footnotes to Your Listing

Up to 10 titles and up to 10 footnotes may be added to your data displays.

The more titles and footnotes on your page, the less space available for the data display, so use of many titles and footnotes is not recommended.

Titles and footnotes may be cancelled by issuing a null title or footnote statement:

```
Title;  
Footnote;
```

Once issued, these have the effect of cancelling all previous title and footnote statements.

These have no effect on any new titles and footnotes subsequently created in your program.

## 14.7 Changing the Order of Your Listing

## 14.8 Sorting by More Than One Variable

If you want your data displayed in a particular order, you can use PROC SORT to re-sort the data.

By default, data are sorted in ascending order (smallest to largest).

So missing values are printed first.

Adding the DESCENDING option in front of a variable name on the BY statement will reverse the sort order, so that the data are sorted in descending order from largest to smallest.

The descending option only affects the single variable following the word DESCENDING.

So the word DESCENDING must be placed in front of each variable that you want sorted in descending order.

Syntax:

```
proc sort;  
  by variablename1 DESCENDING variablename2...variablename'n;  
run;
```

Note: Since the sort procedure is destructive and overwrites the existing data set, it is almost always good programming practice to use an OUT= option when sorting a permanent data set so that the sorted data are written to a temporary data set.

In this class you are not (able) to write data back to any library except WORK, so this necessitates the use of the OUT= option every time you wish to sort a permanent data set.

The sample code from the book has been modified to do this.

The data set LEARN.BLOOD has missing values for the variable CHOL. Sorting by descending CHOL causes the missing values to appear at the bottom of the sorted data.

```
proc sort data=learn.blood out=blood;
  by descending chol;
run;
```

The screenshot displays the SAS Enterprise Guide interface. The main window shows the 'BLOOD (read-only)' dataset, which has been sorted by the 'Chol' variable in descending order. The data is presented in a table with columns: Gender, BloodType, AgeGroup, Subject, WBC, RBC, and Chol. The 'Chol' column contains several missing values (represented by dots) at the bottom of the sorted list.

	Gender	BloodType	AgeGroup	Subject	WBC	RBC	Chol
784	Male	A	Young	970	6130	5.94	99
785	Female	A	Old	426	7220	6.81	97
786	Female	B	Old	776	5840	5.42	96
787	Male	O	Old	930	6550	6.07	96
788	Male	O	Young	987	6020	.	94
789	Male	A	Old	739	6460	4.99	90
790	Male	AB	Old	47	5540	5.27	80
791	Female	AB	Young	79	.	4.61	69
792	Male	O	Old	841	.	3.87	65
793	Female	O	Old	133	8320	4.88	56
794	Male	A	Young	492	.	3.94	36
795	Male	A	Old	829	7950	.	17
796	Male	AB	Old	2	6560	4.7	.
797	Male	B	Old	4	6680	6.85	.
798	Female	O	Young	10	7710	5.55	.
799	Male	O	Young	13	5780	4.37	.
800	Male	A	Old	17	.	5.63	.
801	Female	O	Young	23	7660	4.91	.
802	Female	B	Young	24	8280	6.14	.
803	Female	A	Old	28	7040	3.8	.
804	Male	B	Young	31	5770	7.17	.
805	Male	O	Old	45	7470	5.06	.
806	Female	O	Old	49	.	6.38	.
807	Male	A	Old	60	6680	5.2	.
808	Male	A	Young	66	6580	6.34	.
809	Female	B	Old	68	7560	7.28	.
810	Male	O	Old	70	6680	.	.
811	Male	A	Young	73	7250	6.46	.
812	Male	O	Young	75	7690	7.05	.

The 'Task Status' window at the bottom is empty, showing columns for Task, Status, Queue, and Server.

## 14.9 Labeling Your Column Headings

By default, labels on variables are not displayed in PROC PRINT output.

To enable printing of labels, one of two options must be used on the PROC PRINT statement:  
(1) the keyword LABEL or (2) the SPLIT=*'splitchar'* option.

If any of the variables have permanent labels, all labels will be printed.

To add or change a label on a variable, use a LABEL statement within the PROC PRINT procedure.

Syntax:

```
proc print label;  
  var variablename1 variablename2...variablename'n;  
  label variablename1='this is the label'  
        variablename2='this is a new label'  
        variablename3= ' '; /* inserting a blank between the quotes will cause the  
                               variable to print out without a label */  
run;
```

This is illustrated in Program 14-11.

The `SPLIT = 'splitchar'` option provides a way to control the breakpoints when printing variables with long labels. By default, SAS will use blanks to break long labels for printing. By inserting a break character in the label definition and then specifying this character in the `SPLIT=` definition on the `PROC PRINT` line, you can control the exact breakpoints. Any character can be specified as the split character.

Syntax:

```
proc print split='%';  
  var variablename1 variablename2...variablename'n';  
  label variablename1='this is%the label'  
        variablename2='this%is a%new label';  
run;
```

In this example, `!` is used as the split character.

```
title "Using the SPLIT= option";  
proc print data=sales split='!';  
  id EmpID;  
  var TotalSales Quantity;  
  label EmpID = "Corporate!Employee ID"  
        TotalSales = "Total Sales!Worldwide"  
        Quantity = "Number!Sold!Worldwide";  
  format TotalSales dollar10.2 Quantity comma7.;  
run;
```

You may wish to change all the '!' in the variable labels to blanks and rerun the code to see where SAS will chose to break the labels.

The screenshot shows the SAS Enterprise Guide interface. The Project Explorer on the left lists the project structure, including 'chapter 14' and its sub-items: 'Log', 'Listing - chapter 14', 'HTML - chapter 14', 'PDF - chapter 14', and 'RTF - chapter 14'. The main window displays a table titled 'Using the SPLIT= option' with the following data:

Corporate Employee ID	Total Sales Worldwide	Number Sold Worldwide
0017	\$19,995.00	500
0017	\$1,995.00	100
0177	\$20,000.00	2
0177	\$899.00	100
0177	\$699.00	100
0177	\$52.50	5
1843	\$15,597.00	3
1843	\$15,597.00	3
1843	\$9,109.00	1
1843	\$5,129.00	1
1843	\$599.00	100
1843	\$449.50	50
1843	\$299.50	10
9888	\$1,990.00	1,000
9888	\$299.50	50

The Task Status window at the bottom shows a table with columns: Task, Status, Queue, and Server. The status bar at the bottom indicates 'Ready' and 'justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation'.

## 14.10 Adding Subtotals and Totals to Your Listing

Using a BY statement in PROC PRINT causes SAS to generate a specially formatted report. Using both a BY statement and an ID statement generates another type of report. Finally, adding a SUM statement, adds totals and subtotals to the output.

Syntax:

```
proc print;  
  by variablename1 variablename2...variablename'n';  
  id variablename1 variablename2...variablename'n';  
  var variablename1 variablename2...variablename'n';  
  sum variablename1 variablename2...variablename'n';  
run;
```

When using a BY statement, the data must always be sorted first.

```
proc sort data=learn.sales out=sales;  
  by Region;  
run;  
  
title PROC PRINT with a BY statement and no ID statement;  
proc print data=sales label;  
  by Region;  
  var EmpID TotalSales Quantity;  
  label EmpID = "Employee ID"  
        TotalSales = "Total Sales"  
        Quantity = "Number Sold";  
  format TotalSales dollar10.2 Quantity comma7.;  
run;
```

When used in any procedure, a BY statement always generates output with dashed lines in the Listing output. The dashed lines are absent in the HTML, PDF, and RTF output.

The screenshot displays the SAS Enterprise Guide interface. The main window shows the output of a PROC PRINT statement with a BY statement. The output is organized into three sections, one for each region: East, North, and South. Each section contains a table with columns for Observation (Obs), Employee ID, Total Sales, and Number Sold. The data is separated by dashed lines. The Project Explorer on the left shows the project structure, including a process flow and various output files (Log, HTML, PDF, RTF). The Task Status window at the bottom shows the current task and its status. The status bar at the very bottom indicates the user is logged in as justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation, and the cursor is at Line 1, Col 1.

Obs	Employee ID	Total Sales	Number Sold
Region=East			
1	0177	\$699.00	100
2	0177	\$899.00	100
3	0017	\$19,995.00	500
4	0177	\$20,000.00	2
Region=North			
5	1843	\$449.50	50
6	1843	\$15,597.00	3
7	1843	\$5,129.00	1
8	0177	\$52.50	5
9	1843	\$15,597.00	3
Region=South			
Obs	Employee ID	Total Sales	Number Sold



Adding an ID statement removes the Obs column

The screenshot shows the SAS Enterprise Guide interface. The main window displays the output of a SAS program, titled "Using Labels as Column Headings". The output consists of three tables, one for each region: East, North, and South. Each table has three columns: Employee ID, Total Sales, and Number Sold. The data is as follows:

Region=East		
Employee ID	Total Sales	Number Sold
0177	\$699.00	100
0177	\$899.00	100
0017	\$19,995.00	500
0177	\$20,000.00	2

Region=North		
Employee ID	Total Sales	Number Sold
1843	\$449.50	50
1843	\$15,597.00	3
1843	\$5,129.00	1
0177	\$52.50	5
1843	\$15,597.00	3

Region=South		
Employee ID	Total Sales	Number Sold

The Task Status window at the bottom shows a table with columns: Task, Status, Queue, and Server.

When you use a SUM statement and a BY statement with one BY variable, PROC PRINT sums the SUM variables for each BY group that contains more than one observation and totals them over all BY groups.

This is illustrated in Program 14-13.

### 14.11 Making Your Listing Easier to Read

Using the same variables on both the BY statement and ID statement generates a nicely formatted report.

Only the first occurrence of each BY variable is printed.

In the Listing output, when SAS reaches the end of a BY group, it also skips a line before starting to print the next BY group.

This is illustrated in Program 14-14.

### 14.12 Adding the Number of Observations to Your Listing

Adding the 'N' option on the PROC PRINT statement will cause the total number of observations to be printed at the end of your output (N = *number of observations*).

Syntax:

```
proc print n;  
  var variablename1 variablename2...variablename'n';  
run;
```

Here is the output from Program 14-15. Without any text specified, SAS prints “N” on the left side of the =

The screenshot shows the SAS Enterprise Guide interface. The main window displays the output of a SAS program. The title bar indicates the window is titled "Listing - chapter 14". The output is a table with the following data:

Emp ID	TotalSales	Quantity
0017	\$19,995.00	500
0017	\$1,995.00	100
0177	\$699.00	100
0177	\$899.00	100
0177	\$52.50	5
0177	\$20,000.00	2
1843	\$449.50	50
1843	\$599.00	100
1843	\$15,597.00	3
1843	\$5,129.00	1
1843	\$299.50	10
1843	\$9,109.00	1
1843	\$15,597.00	3
9888	\$299.50	50
9888	\$1,990.00	1,000

Below the table, the output shows "N = 15".

The Project Explorer on the left shows a project structure with a "chapter 14" folder containing "Log", "Listing - chapter 14", "HTML - chapter 14", "PDF - chapter 14", and "RTF - chapter 14".

The Task Status window at the bottom shows a table with columns "Task", "Status", "Queue", and "Server".

The status bar at the bottom indicates "Ready" and "justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation".

A label can also be added by using the format: N=*'label you specify'*

Syntax:

```
proc print n='Number of Observations is:';  
  var variablename1 variablename2...variablename'n';  
run;
```

This is illustrated in Program 14-15 on page 279.

### 14.13 Double-Spacing Your Listing

To double-space your output, use the DOUBLE option on the PROC PRINT statement.

This only has an effect on the output in the LISTING window. It has no impact on the HTML, PDF, and RTF output.

Syntax:

```
proc print double;  
  var variablename1 variablename2...variablename'n';  
run;
```

## 14.14 Listing the First n Observations of Your Data Set

To list the first 'n' observations in a data set, use the OBS= option after the data set name on the PROC PRINT statement.

To start the list at a specified observation, use the FIRSTOBS= option.

To select a specified number of observations starting at a specific observation use both options: FIRSTOBS= to define the first observation and OBS= to define the last observation

Syntax:

```
proc print data=datasetname(obs=10); /* prints observations 1-10 */  
proc print data=datasetname(firstobs=10); /* prints all observations except 1-9 */  
proc print data=datasetname(firstobs=10 obs=25); /* prints observations 10-25 */
```

In the Listing output, PROC PRINT will attempt to print all variables on a single output line by reducing the amount of white space between variable columns and printing variable names vertically.

To eliminate the possibility of generating output with variable names listed vertically, use the HEADING=H option on the PROC PRINT statement.

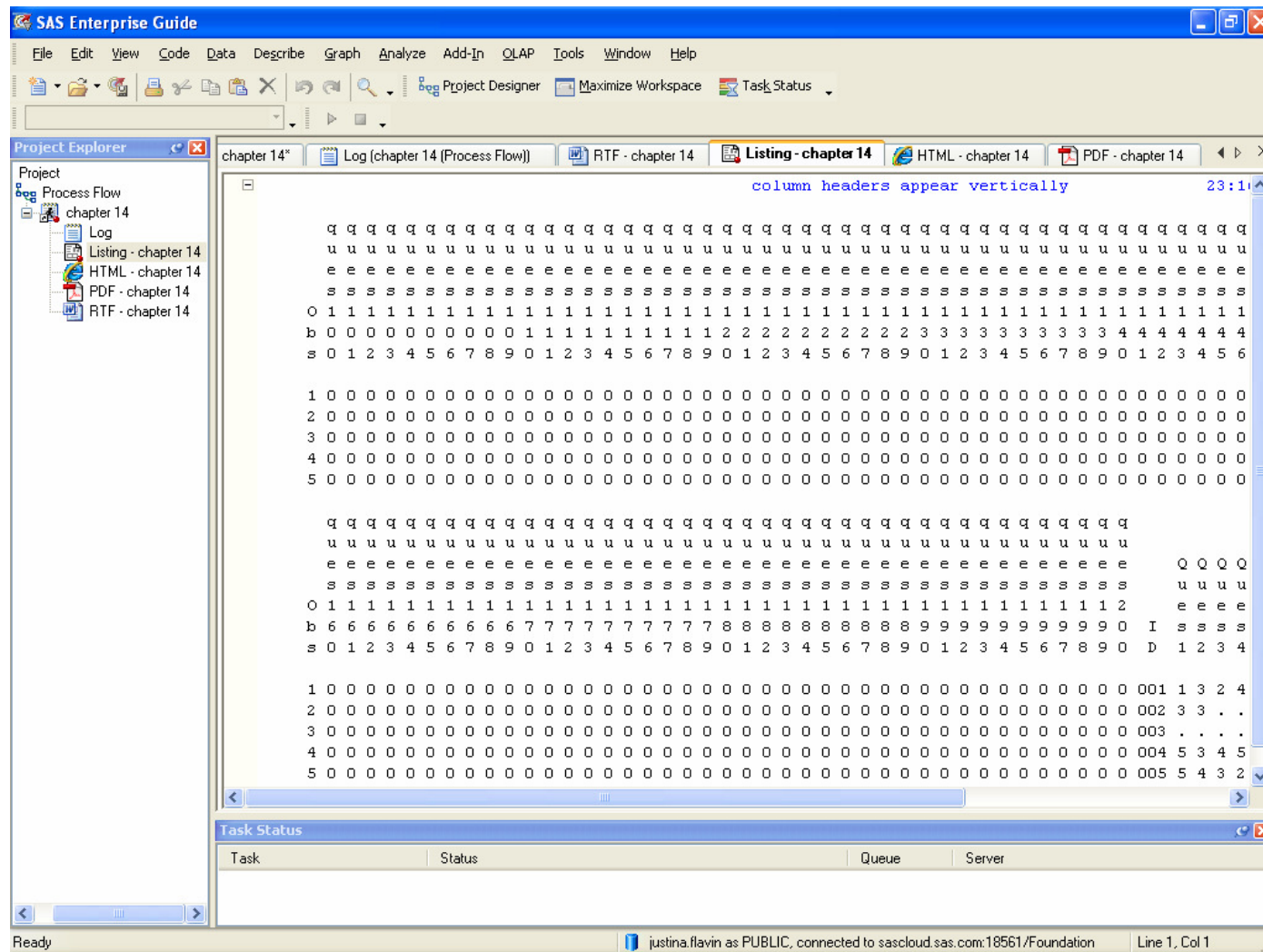
Syntax:

```
proc print heading=h;  
  var variablename1 variablename2...variablename'n';  
run;
```

This is best illustrated in a data set with many variables.

```
/* Create a data set with many variables */  
data manyvars;  
  retain ques100-ques200 0;  
  set learn.psych;  
run;  
  
proc print;  
  title 'column headers appear vertically';  
run;
```

This generates output that is not easy to read or understand.



Adding the heading=h option generates output that is easier to review.

```
proc print heading=h;
```

```
  title 'column headers appear horizontally';
```

```
run;
```

This option has no effect on the HTML, PDF, or RTF output.

The screenshot displays the SAS Enterprise Guide interface. The main window shows the output of a PROC PRINT statement. The output is formatted with horizontal column headers, as specified by the heading=h option. The output is organized into four sections, each starting with an 'Obs' label followed by a list of question identifiers (ques100 through ques155). Each section contains five rows of data, with the first row being the header row and the subsequent four rows being data rows. The data values are all 0. The interface includes a Project Explorer on the left, a Task Status window at the bottom, and a status bar at the very bottom.

Obs	ques100	ques101	ques102	ques103	ques104	ques105	ques106	ques107	ques108	ques109	ques110	ques111
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0

Obs	ques115	ques116	ques117	ques118	ques119	ques120	ques121	ques122	ques123	ques124	ques125	ques126
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0

Obs	ques130	ques131	ques132	ques133	ques134	ques135	ques136	ques137	ques138	ques139	ques140	ques141
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0

Obs	ques145	ques146	ques147	ques148	ques149	ques150	ques151	ques152	ques153	ques154	ques155	ques156
1	0	0	0	0	0	0	0	0	0	0	0	0



This chapter has covered most of the statements that are available in PROC PRINT.

For additional information refer to the online documentation.

The online documentation also contains many helpful and more sophisticated examples with code that you can cut and paste into your SAS session.