

SAS Programming (BIOL-4V190)

Chapter 15 Creating Customized Reports

15.1 Introduction

PROC REPORT is a procedure that can generate basic data displays like PROC PRINT, but it has features and controls that enable you to create more sophisticated output as well.

Note: All of the code as presented in this lecture works as described on the output in the Listing window. There may be some differences in the output that appears in the HTML, PDF, and RTF windows.

15.2 Using PROC REPORT

PROC REPORT is an interactive procedure by default, so once you submit and run the code, an interactive editor window opens up.

To turn off this feature, the option NOWINDOWS (or NOWD) is used on the PROC REPORT line.

In this course, we will not be using the interactive editor window, so be sure to always include NOWD in your code.

By default, variable labels (if they exist) are used as the column headers in PROC REPORT.

This is opposite to PROC PRINT, where the default is to use the variable names as the column headers.

This can be seen when comparing the PROC PRINT and PROC REPORT output in Programs 15-1 and 15-2.

Syntax:

```
proc report nowd;  
run;
```

15.3 Selecting Variables to Include in Your Report

To select the variables that should be included in the output, use the COLUMN (or COLUMNS) statement.

Syntax:

```
proc report nowd;  
  columns variablename1 variablename2...variablename'n';  
run;
```

15.4 Comparing Detail and Summary Reports

PROC REPORT handles the default display of numeric variables and character variables differently than PROC PRINT.

If PROC REPORT is run on a data set that contains only character variables –or– both character and numeric variables (or the COLUMNS statement contains only character variables –or– both character and numeric variables), the resulting output will be a line by line display of all records (like PROC PRINT).

This is called a DETAIL report.

However, if PROC REPORT is run on a data set that contains only numeric variables (or the COLUMNS statement only contains numeric variables), the resulting output will contain a single data line showing the SUM of the values of each variable.

This is called a SUMMARY report.

Within PROC REPORT, variables are assigned a variable type which determines how they will appear in the output.

By default, numeric variables are assigned a type of ANALYSIS with a summary statistic of SUM, while character variables are assigned a type of DISPLAY.

To control and change the type (as well as other variable attributes), the DEFINE statement is used.

The variable name is placed in front of the '/' on the DEFINE statement, while the attribute options are placed behind the '/'.

Attribute options include the variable types, labels, column widths, formats, and data display justification.

The variable type options are DISPLAY, ANALYSIS, GROUP, ORDER, ACROSS, and COMPUTED.

To add a label to a variable, put the label in quotation marks.

To control the width of the column use the width='n' option.

Syntax:

```
proc report nowd;  
  columns variablename1 variablename2...variablename'n';  
  define variablename1 / attribute options;  
  define variablename2 / attribute options;  
run;
```

This example illustrates the default behavior of PROC REPORT when the COLUMN statement contains only numeric variables.

By default, numeric variables are assigned a type of ANALYSIS with a summary statistic of SUM and the resulting output will contain a single data line showing the SUM of the values of each variable.

```
*Program 15-4 Using PROC REPORT with only numeric variables - page 292;  
title "Report with Only Numeric Variables";  
proc report data=learn.medical nowd;  
    column HR Weight;  
run;
```

The output for this example is shown on page 292.

By adding DEFINE statements for each variable, we can control the resulting data display:

```
*Program 15-5 Using DEFINE statements to define a display usage - page 292;  
title "Display Usage for Numeric Variables";  
proc report data=learn.medical nowd;  
    column HR Weight;  
    define HR / display "Heart Rate" width=5;  
    define Weight / display width=6;  
run;
```

Although HR and Weight are numeric variables, they are specified as DISPLAY variables on the DEFINE statement.

A width option can be specified to control the width of the column. Finally, a label placed in quotation marks can be specified to override the variable label that may be stored on the data set.

The output for this example is shown on page 293.

15.5 Producing a Summary Report

The variable type GROUP is used to create a report having the data displayed or summarized by the categories of certain variables (the grouping variables).

This is similar to using the BY or CLASS statement in other procedures.

To change the summary statistic on an ANALYSIS variable, list the statistic keyword after the word ANALYSIS. A list of statistics keywords can be found in the online documentation.

```
*Program 15-6 Specifying a GROUP usage to create a summary report - page 293;
title "Demonstrating a GROUP Usage";
proc report data=learn.medical nowd;
  column Clinic HR Weight;
  define Clinic / group width=11;
  define HR / analysis mean "Average Heart Rate" width=12
              format=5.;
  define Weight / analysis mean "Average Weight" width=12
                  format=6.;
run;
```

In this example, the variable CLINIC is defined as a GROUP variable.

A GROUP variable causes the data to be summarized or collapsed based on the values of the GROUP variable, so that the report contains one row per unique GROUP variable value.

HR and WEIGHT are defined as ANALYSIS variables and the statistic that will be reported is the variable MEAN. So this report will contain one row per unique value of CLINIC and the HR and WEIGHT columns will contain the MEAN value of those variables as calculated for that value of CLINIC.

The output for this example is shown on page 294.

15.6 Demonstrating the FLOW Option of PROC REPORT

To display a long text variable, the FLOW option can be added to the DEFINE statement to enable word wrapping.

To control the breakpoints in the data (and in your labels), the SPLIT=*'split character'* option is added to the PROC REPORT statement.

By default, in PROC REPORT, the split character is '/'.

Thus, if there are backslashes in your data, they will not be displayed and will be interpreted as embedded split characters.

To control the data display justification use RIGHT, LEFT, or CENTER on the DEFINE statement.

This option will right justify, left justify, or center the data under the column header.

To add a line separator under the column headers, use the HEADLINE option on the PROC REPORT statement.

To add a blank line between the column headers and the first line of data, use the HEADSKIP option on the PROC REPORT statement.

To control the width of your output, use the LINESIZE=*'n'* (or LS=*'n'*) option on the PROC REPORT statement.

To control the number of data lines to display on a page, use the PAGESIZE=*'n'* (or PS=*'n'*) option on the PROC REPORT statement.


```

*Program 15-7 Demonstrating the FLOW option with PROC REPORT - page 294;
title "Demonstrating the FLOW Option";
proc report data=learn.medical nowd headline
      split=' ' ls=74;
  column Patno VisitDate DX HR Weight Comment;
  define Patno      / "Patient Number" width=7;
  define VisitDate  / "Visit Date" width=9 format=date9.;
  define DX         / "DX Code" width=4 right;
  define HR         / "Heart Rate" width=6;
  define Weight     / width=6;
  define Comment    / width=30 flow;
run;

```

This example illustrates the use of the FLOW option to control the display of a long comment variable.

The SPLIT= option is used to specify a split character (a blank in this case) for the column labels.

The RIGHT option is specified for the variable DX to right justify the data in that column.

The HEADLINE option places a line underneath the column headers.

The output for this example is shown on page 295.

15.7 Using Two Grouping Variables

Just as more than one variable can be specified on a BY or CLASS statement, more than one GROUP variable can be defined in PROC REPORT.

```
*Program 15-9 Demonstrating the effect of two variables with GROUP usage - page 296;
title "Multiple GROUP Usages";
proc report data=learn.bicycles nowd headline ls=80;
  column Country Model Units TotalSales;
  define Country / group width=14;
  define Model / group width=13;
  define Units / sum "Number of Units" width=8
               format=comma8.;
  define TotalSales / sum "Total Sales (in thousands)"
                       width=15 format=dollar10.;
run;
```

In this example, the data will be summarised for each unique combination of Country and Model that exists in the data.

Units and TotalSales are ANALYSIS variables and the summary statistic to be reported for these variables is SUM, or the sum of the values.

This output is shown on page 297.

15.8 Changing the Order of Variables in the COLUMN Statement

Changing the order of the variables on the COLUMN statement changes the order in which they appear in the output.

The DEFINE statements DO NOT need to be listed in the same order as the variables on the COLUMN statement.

15.9 Changing the Order of Rows in a Report

15.10 Applying the ORDER Usage to Two Variables

The variable type ORDER is used to create a report having the data displayed in ascending sorted order. To change the sort order to descending sorted order, use the keyword DESCENDING in front of ORDER. The data do not need to be sorted (with PROC SORT) prior to using ORDER in PROC REPORT.

```
*Program 15-12 Applying the ORDER usage for two variables - page 300;  
title "Applying the ORDER Usage for Two Variables";  
proc report data=learn.sales nowd headline;  
    column EmpID Quantity TotalSales;  
    define EmpID / order "Employee ID" width=11;  
    define TotalSales / descending order "Total Sales"  
                        width=9 format=dollar9.;  
    define Quantity / width=8 format=comma8.;  
run;
```

In this example, EmpID is defined as an ORDER variable.

The data will be displayed in ascending order of EmpID.

Similar to using VAR and ID statements with identical variables in PROC PRINT, only the first occurrence of each unique value of EmpID is printed.

TotalSales is also an ORDER variable and by the addition of the DESCENDING option, the data will be displayed in DESCENDING order of TOTALSALES.

The output is shown on page 301.

15.11 Creating a Multi-Column Report

To create a multi-column report (like a newspaper or a telephone book), use the PANELS='n' option on the PROC REPORT statement where n is the number of panels (or columns) to display per page.

This is illustrated in Program 15-13.

15.12 Producing Report Breaks

To produce output with totals and subtotals on analysis variables, use the RBREAK and BREAK statements.

Both statements must be followed with the keywords BEFORE or AFTER.

Following this, BREAK also requires the name of an ORDER or GROUP variable.

RBREAK BEFORE places a summary statistic line at the beginning of the report.

RBREAK AFTER places a summary statistic line at the end of the report.

BREAK BEFORE *variablename* generates a summary statistic line each time the value of *variablename* changes and is displayed at the top of subset of data that was used to calculate it.

BREAK AFTER *variablename* generates a summary statistic line each time the value of *variablename* changes and is displayed at the bottom of the subset of data that was used to calculate it.

Options that can be placed after the '/' include:

OL – prints a single line above the summary statistic line (overline)

DOL – prints a double line above the summary statistic line

UL – prints a single line below the summary statistic line (underline)

DUL – prints a double line below the summary statistic line

SKIP – places a blank line after the summary line

SUMMARIZE – prints a summary statistic

SUPPRESS – prevents printing of the BREAK variable on the summary line

```

*Program 15-14 Requesting a report break (RBREAK statement) - page 303;
title "Producing Report Breaks";
proc report data=learn.sales nowd headline;
  column Region Quantity TotalSales;
  define Region / width=6;
  define Quantity / sum width=8 format=comma8.;
  define TotalSales / sum "Total Sales" width=9
                      format=dollar9.;
  rbreak after / dol dul summarize;
run;

```

In this example, RBREAK AFTER places a summary statistic line at the end of the report.

DOL creates a double line above the summary statistic line.

DUL – prints a double line below the summary statistic line.

SUMMARIZE – prints a summary statistic, in this case, SUM.

This output is shown on page 304.

```

*Program 15-15 Demonstrating the BREAK statement of PROC REPORT - page 304;
title "Producing Report Breaks";
proc report data=learn.sales nowd headline;
    column Region Quantity TotalSales;
    define Region / order width=6;
    define Quantity / sum width=8 format=comma8.;
    define TotalSales / sum "Total Sales" width=9
                        format=dollar9.;
    break after region / ol summarize skip;
run;

```

This example is similar to the previous example.

The BREAK AFTER REGION statement instructs SAS to generate a summary statistic line each time the value of REGION changes.

The value of REGION is also displayed at the bottom of the subset of data that was used to calculate it.

This report is shown on page 305.

15.13 Using a Nonprinting Variable to Order a Report

PROC REPORT has a unique feature that allows you to create a data display sorted by the values of a variable that do not appear in the output.

The variable must be included on the COLUMNS statement and a DEFINE statement must be used.

The variable must be defined as a GROUP variable, and the NOPRINT option must be specified.

*Program 15-16 Using a non-printing variable to order the rows of a report - page 306;

```
data temp;
    set learn.sales;
    length LastName $ 10;
    LastName = scan(Name,-1,' ');
run;

title "Listing Ordered by Last Name";
proc report data=temp nowd headline;
    column LastName Name EmpID TotalSales;
    define LastName / group noprint;
    define Name / group width=15;
    define EmpID / "Employee ID" group width=11;
    define TotalSales / sum "Total Sales" width=9
                        format=dollar9.;
run;
```

In this example, a new variable, LastName, containing just the person's last name is created.

Then this variable is listed on the COLUMN statement to make it available to the procedure.

The DEFINE statement has the NOPRINT option, so that the variable values can be used to GROUP the observations by LastName, but the variable does not appear on the printed output.

This output is shown on page 307.

15.14 Computing a New Variable with PROC REPORT

Within PROC REPORT, new variables can be defined, calculated and displayed.

The new variable name is placed on the COLUMNS statement and calculated within a COMPUTE block.

On the DEFINE statement, the variable type is COMPUTED.

The COMPUTE block is placed after the DEFINE statement.

Syntax:

```
compute variablename;  
    programming statements;  
endcomp;
```

```
*Program 15-17 Computing a new variable with PROC REPORT - page 307;
```

```
title "Computing a New Variable";
```

```
proc report data=learn.medical nowd;
```

```
    column Patno Weight WtKg;
```

```
    define Patno / display "Patient Number" width=7;
```

```
    define Weight / display noprint width=6;
```

```
    define WtKg / computed "Weight in Kg"  
                  width=6 format=6.1;
```

```
    compute WtKg;
```

```
        WtKg = Weight / 2.2;
```

```
    endcomp;
```

```
run;
```

In this example, WtKg is calculated in a COMPUTE block.

WtKg is also included on the COLUMN statement and has its own DEFINE statement.

WtKg is calculated using the value of Weight as input, so although Weight is not included in the final report, Weight must be included in the COLUMN statement to make it available in the procedure.

15.15 Computing a Character Variable in a COMPUTE Block

Creating a character variable within PROC REPORT is similar to creating a numeric variable. However, the COMPUTE statement must also specify the variable type CHARACTER.

```
title "Creating a Character Variable in a COMPUTE Block - page 309";  
proc report data=learn.medical nowd;  
  column Patno HR Weight Rate;  
  define Patno / display "Patient Number" width=7;  
  define HR / display "Heart Rate" width=5;  
  define Weight / display width=6;  
  define Rate / computed width=6;  
  compute Rate / character length=6;  
    if HR gt 75 then Rate='Fast';  
    else if HR gt 55 then Rate='Normal';  
    else if not missing(HR) then rate='Slow';  
  endcomp;  
run;
```

The output for this example is shown on page 309.

15.16 Creating an ACROSS Variable with PROC REPORT

To create a report with nested variables, variable names are listed on the COLUMNS statement with a comma between the variables.

The variables will be stacked in the report in the order in which they appear in the COLUMNS statement.

Variables which precede a comma are defined as ACROSS variables in the DEFINE statement.

The ACROSS variable name or label will be centered above the columns it spans.

```
*Program 15-18 Demonstrating an ACROSS usage in PROC REPORT - page 310;
***Demonstrating an Across Usage;
title "Demonstrating an ACROSS Usage";
proc report data=learn.bicycles nowd headline ls=80;
  column Model,Units Country;
  define Country      / group width=14;
  define Model        / across "Model";
  define Units        / sum "# of Units" width=14
                      format=comma8.;
run;
```

In this example, Model is an ACROSS variable.

This means that the values of Model will not be placed in a column within the report, but will span across the report horizontally.

Units will be summed for each unique value of Model.

The output is shown on page 310.

15.17 Modifying the Column Label for an ACROSS Variable

If the label or variable name of an ACROSS variable begins and ends with certain characters such as `_*<>`, SAS will treat those characters as expansion characters.

The variable name or label will still be centered above the columns it spans and the expansion characters will be repeated before and after the variable name to cover the width of all of the spanned columns.

This is illustrated in the output on page 311.

15.18 Using an ACROSS Usage to Display Statistics

If an ACROSS variable spans analysis variables, the ACROSS variable functions like a GROUP variable and causes the statistics to be generated within the categories of the ACROSS variable.

```
title "Average Blood Counts by Age Group - page 312";
proc report data=learn.blood nowd headline;
  column Gender BloodType AgeGroup,WBC AgeGroup,RBC;
  define Gender / group width=8;
  define BloodType / group width=8 "Blood Group";
  define AgeGroup / across "- Age Group -";
  define WBC / analysis mean format=comma8.;
  define RBC / analysis mean format=8.2;
run;
```

In this example, AgeGroup is an ACROSS variable spanning WBC and RBC. This causes mean values of RBC and WBC to be calculated within the two categories of AgeGroup. This is similar to using a BY or CLASS statement in PROC MEANS.

The output is shown on page 312.

*****Working with Missing Values in GROUP, ORDER, and ACROSS Variables*****

By default, when there are missing values in GROUP, ORDER, or ACROSS variables, these will not be included or displayed in the output.

To include missing values, use the MISSING option on the PROC REPORT statement.

In the code file for this chapter, you will find the code to create the data set GROMISS.

There are missing values in the data in GROCMISS.

PROC REPORT is run twice, with and without the MISSING option.

The results are very different.

```
proc report data=grocmis nowd headline;  
  column sector manager sales;  
  define sector / group;  
  define manager / group;  
  define sales / format=dollar9.2;  
  rbreak after / dol summarize;  
  title 'Summary Report for All Sectors and Managers - without MISSING option';  
run;
```

Without the MISSING option, the observations with missing values for Sector and Manager (the GROUP variables) ARE NOT included in the output and the calculation of Total.

The screenshot shows the SAS Enterprise Guide interface. The main window displays a summary report titled "Summary Report for All Sectors and Managers - without MISSING option". The report includes a table with the following data:

Sector	Manager	Sales
ne	7	\$596.00
	8	\$1,045.00
nw	3	\$690.00
	4	\$598.00
	9	\$746.00
se	1	\$130.00
	2	\$630.00
sw	5	\$353.00
	6	\$655.00
		\$5,443.00

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

Now the MISSING option is added to the PROC REPORT statement.

```
proc report data=grocmiss nowd headline missing;  
  title 'Summary Report for All Sectors and Managers - with MISSING option';  
  column sector manager sales;  
  define sector / group;  
  define manager / group;  
  define sales / format=dollar9.2;  
  rbreak after / dol summarize;  
run;
```

With the MISSING option, the observations with missing values for Sector and Manager (the GROUP variables) ARE included in the output and the calculation of Total.

The screenshot shows the SAS Enterprise Guide interface. The main window displays a report titled "Summary Report for All Sectors and Managers - with MISSING option". The report includes a table with the following data:

Sector	Manager	Sales
.	.	\$40.00
.	1	\$100.00
.	3	\$420.00
ne	.	\$190.00
.	7	\$596.00
.	8	\$1,045.00
nw	3	\$690.00
.	4	\$598.00
.	9	\$746.00
se	.	\$120.00
.	1	\$130.00
.	2	\$630.00
sw	5	\$353.00
.	6	\$655.00
.		\$6,313.00

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

Controlling the Output Order With the ORDER= Option in GROUP, ORDER, and ACROSS Variables

By default, the formatted values are used to determine the print order for GROUP, ORDER, and ACROSS variables.

This can be changed through the use of the ORDER= option on the DEFINE statement.

There are 4 possible values for the ORDER= statement:

ORDER=FORMATTED	values ordered by formatted values (default)
ORDER=INTERNAL	values ordered by internal values
ORDER=FREQ	values ordered by frequency counts of each value
ORDER=DATA	values ordered by physical order in the data set

This is similar to using the ORDER= option in PROC FREQ (Section 17.8 on page 353).

In the code file for this chapter, formats are created and applied to the data in GROMISSF.

```
proc report data=grocmissf nowd headline missing;
  title 'by default, values of the formatted values are used to order the output';
  column sector manager department sales;
  define sector / group;
  define department / group;
  define manager / group;
  define sales / format=dollar9.2;
  rbreak after / dol summarize;
run;
```

The variables are displayed by the formatted values.

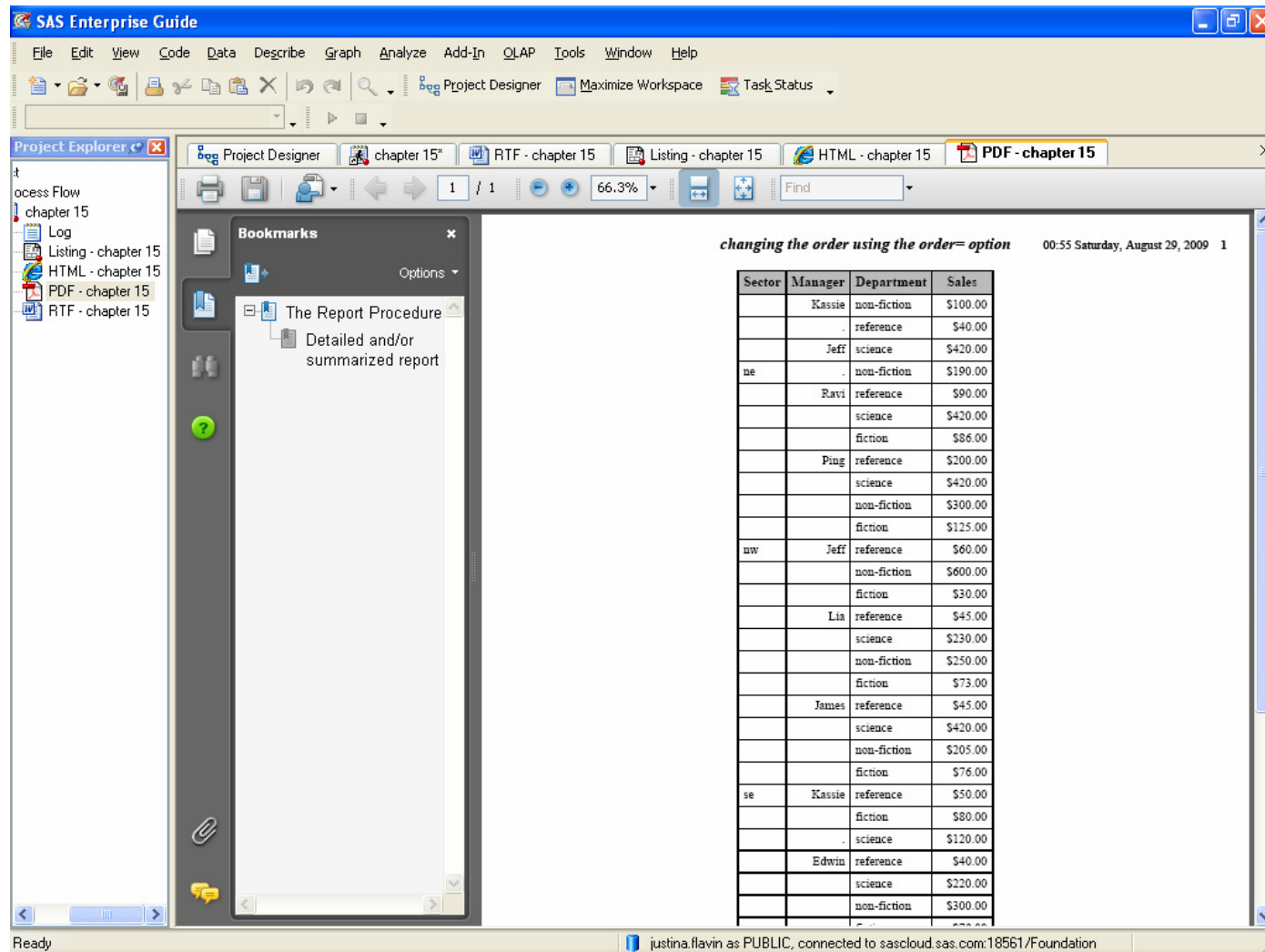
The screenshot displays the SAS Enterprise Guide software interface. The main window shows a report titled "by default, values of the formatted values are used to order the output". The report is displayed in a table format with the following columns: Sector, Manager, Department, and Sales. The data is sorted by the Sales column in descending order. The table contains 24 rows of data, including managers like Jeff, Kassie, Ping, Ravi, James, Lia, Edwin, and Kassie, across various departments like reference, science, non-fiction, and fiction. The interface also shows a Project Explorer on the left, a Bookmarks pane, and a status bar at the bottom indicating the user is justina.flavin as PUBLIC, connected to sascloud.sas.com:18561/Foundation.

Sector	Manager	Department	Sales
.	.	reference	\$40.00
.	Jeff	science	\$420.00
.	Kassie	non-fiction	\$100.00
ne	.	non-fiction	\$190.00
.	Ping	fiction	\$125.00
.	.	non-fiction	\$300.00
.	.	reference	\$200.00
.	.	science	\$420.00
.	Ravi	fiction	\$86.00
.	.	reference	\$90.00
.	.	science	\$420.00
nw	James	fiction	\$76.00
.	.	non-fiction	\$205.00
.	.	reference	\$45.00
.	.	science	\$420.00
.	Jeff	fiction	\$30.00
.	.	non-fiction	\$600.00
.	.	reference	\$60.00
.	Lia	fiction	\$73.00
.	.	non-fiction	\$250.00
.	.	reference	\$45.00
.	.	science	\$230.00
se	.	science	\$120.00
.	Edwin	fiction	\$70.00
.	.	non-fiction	\$300.00
.	.	reference	\$40.00
.	.	science	\$220.00
.	Kassie	fiction	\$80.00

In this example, the ORDER=INTERNAL option is used for DEPARTMENT, and ORDER=DATA for Manager.

```
proc report data=grocmissf nowd headline missing;  
    title 'changing the order using the order= option';  
    column sector manager department sales;  
    define sector / group;  
    define department / order=internal group;  
    define manager / order=data group;  
    define sales / format=dollar9.2;  
    rbreak after / dol summarize;  
run;
```

Sector is still displayed by ascending order of the formatted values. Manager and Department are displayed based on the data value position and data value in the raw (unformatted) data.



SAS Enterprise Guide

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

Project Explorer

- Process Flow
- chapter 15
 - Log
 - Listing - chapter 15
 - HTML - chapter 15
 - PDF - chapter 15
 - RTF - chapter 15

Project Designer

chapter 15* RTF - chapter 15 Listing - chapter 15 HTML - chapter 15 PDF - chapter 15

1 / 1 66.3% Find

Bookmarks

- The Report Procedure
 - Detailed and/or summarized report

changing the order using the order= option 00:55 Saturday, August 29, 2009 1

Sector	Manager	Department	Sales
	Kassie	non-fiction	\$100.00
		reference	\$40.00
	Jeff	science	\$420.00
ne		non-fiction	\$190.00
	Ravi	reference	\$90.00
		science	\$420.00
		fiction	\$86.00
	Ping	reference	\$200.00
		science	\$420.00
		non-fiction	\$300.00
		fiction	\$125.00
nw	Jeff	reference	\$60.00
		non-fiction	\$600.00
		fiction	\$30.00
	Lia	reference	\$45.00
		science	\$230.00
		non-fiction	\$250.00
		fiction	\$73.00
	James	reference	\$45.00
		science	\$420.00
		non-fiction	\$205.00
		fiction	\$76.00
se	Kassie	reference	\$50.00
		fiction	\$80.00
		science	\$120.00
	Edwin	reference	\$40.00
		science	\$220.00
		non-fiction	\$300.00

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