(6) PROCEDURES

Up to now, SAS procedures have been used without much discussion as to how they work, options available, or why you would use one procedure in lieu of another. There has been a lot of explanation of how to convert raw data into SAS data sets, how to read SAS data sets, the rules for naming variables and data sets, etc. without much reference to using data sets in procedures. All of the discussion of SAS rules and SAS statements will have more grounding if put into the context of what you can do with data once it is in a SAS data set.

As mentioned in the introduction, there many parts to SAS and there are procedures that are specific to the various parts of SAS. If you perform a regression analysis using your data, you will have to use a procedure that is part of SAS/STAT. If you want to create presentation quality graphics, you will have to use a procedure that is part of SAS/GRAPH. Many of the main tasks you want to perform with data are accomplished using BASE SAS procedures: rearranging data; reporting; counting; computing descriptive statistics. These tasks are demonstrated using PROC PRINT and PROC REPORT (reporting), PROC FREQ and PROC TABULATE (counting), PROC SORT (rearranging), PROC MEANS, PROC UNIVARIATE, PROC TABULATE, and PROC REPORT (descriptive statistics).

The data to be used in discussing these procedures are in a data set named CANCER99. There are five variables in that data set and they are shown below in the output from PROC CONTENTS.

```
                          CONTENTS OF DATA SET CANCER99

Data Set Name        X.CANCER99                              Observations          37583
Member Type          DATA                                    Variables             5
Engine               V9                                      Indexes               0
Created              Thursday, October 04, 2007 04:43:47 PM  Observation Length    16
Last Modified        Thursday, October 04, 2007 04:43:47 PM  Deleted Observations  0
Protection                                                   Compressed            NO
Data Set Type                                                Sorted                NO
Label
Data Representation  WINDOWS_32
Encoding             wlatin1  Western (Windows)


                       Engine/Host Dependent Information

             Data Set Page Size          4096
             Number of Data Set Pages    150
             First Data Page             1
             Max Obs per Page            252
             Obs in First Data Page      160
             Number of Data Set Repairs  0
             File Name                   k:\epi514\datasets\cancer99.sas7bdat
             Release Created             9.0101M3
             Host Created                XP_PRO


                  Alphabetic List of Variables and Attributes

                    #    Variable    Type    Len

                    3    age         Num       8
                    4    cause       Char      4
                    1    county      Char      2
                    2    gender      Char      1
                    5    place       Char      1
```

This is the default output from PROC CONTENTS, run with no extra options as follows.

...Example 6.1...

```
libname x 'k:\epi514\datasets';

title 'CONTENTS OF DATA SET CANCER99';
proc contents data=x.cancer99;
run;
```

The default output of PROC CONTENTS gives you details about a data set. In addition to the names and attributes of variables, it also tells you the number of observations and variables in the data set and the date that the data set was created. You can change the order of the list of variables by uisng a VARNUM option. If you have a mix of variables beginning with both small and capital letters, PROC CONTENTS will list all variables starting with capital letters first, followed by those starting with small letters.

...Example 6.2...

```
data test; ①
a_variable = 10;
c_variable = 20;
B_variable = 30;
Z_variable = 'MIKE';
run;

title 'MIXED CASE STARTING LETTERS';
proc contents data=test; ②
run;

title 'MIXED CASE STARTING LETTERS-IGNORECASE PROC OPTION';
proc contents data=test order=ignorecase; ③
run;

options validvarname=upcase; ④

title 'MIXED CASE STARTING LETTERS-VALIDVARNAME OPTION';
proc contents data=test; ⑤
run;

options validvarname=v7; ⑥
```

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	B_variable	Num	8
4	Z_variable	Char	4
1	a_variable	Num	8
2	c_variable	Num	8

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	a_variable	Num	8
3	B_variable	Num	8
2	c_variable	Num	8
4	Z_variable	Char	4

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	A_VARIABLE	Num	8
3	B_VARIABLE	Num	8
2	C_VARIABLE	Num	8
4	Z_VARIABLE	Char	4

A small data set is created that contains variables that start with both small and capital letters ①. PROC CONTENTS run without any options produces a variable list that has the capital letters first, followed by the small letters (upper box on the above right) ②. Adding an IGNORECASE option in PROC CONTENTS places the variables in alphabetic order (middle box on the above right) ③. Setting the VALIDVARNAME system option to UPCASE ④ produces a list of variables in alphabetic order (lower box on the above right) even when PROC CONTENTS is run without an IGNORECASE option ⑤. However, all the variables are displayed in uppercase. To return to the default display of variables, the VALIDVARNAME system option is reset to V7 ⑥.

The VARNUM option changes the order of the variable list to correspond with the variable position in the data set, while the SHORT option limits the PROC CONTENTS output to a list of variable names that are still, by default, in alphabetic order following all the rules described in example 6.2.

...Example 6.3...

```
libname x "k:\epi514\datasets";

title 'CONTENTS OF DATA SET CANCER99 - VARNUM PROC OPTION';
proc contents data=x.cancer99 varnum;
run;

title 'CONTENTS OF DATA SET CANCER99 - SHORT PROC OPTION';
proc contents data=x.cancer99 short;
run;
```

Variables in Creation Order

#	Variable	Type	Len
1	county	Char	2
2	gender	Char	1
3	age	Num	8
4	cause	Char	4
5	place	Char	1

The output from the above procedures is shown on the right.

CONTENTS OF DATA SET CANCER99 - SHORT PROC OPTION

Alphabetic List of Variables for X.CANCER99

age cause county gender place

Before moving on to using the data set CANCER99 in various procedures, you can use another procedure to change all the variable names to uppercase and to add labels to the data set. SAS data sets have two parts: a header that contains information about the variables and observations in the data set; the actual data . If you want to change the contents of information that resides in the header of the data set, the most efficient method is to use PROC DATASETS. Some of the information that resides in the header includes: variable names; variable labels; if a format is assiociated with one or more variables in the data set.

...Example 6.4...

```
libname x "k:\epi514\datasets";

proc datasets lib=x nolist; ①
modify cancer99; ②
label ③
county = 'GAZETTEER COUNTY NUMBER'
age    = 'AGE AT DEATH (YEARS)'
cause  = 'ICD-10 CAUSE OF DEATH'
place  = 'WHERE DEATH OCCURRED'
;
rename ④
county = COUNTY
cause  = CAUSE
age    = AGE
gender = GENDER
place  = PLACE
;
quit; ⑤
```

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Label
3	AGE	Num	8	AGE AT DEATH (YEARS)
4	CAUSE	Char	4	ICD-10 CAUSE OF DEATH
1	COUNTY	Char	2	GAZETTEER COUNTY NUMBER
2	GENDER	Char	1	
5	PLACE	Char	1	WHERE DEATH OCCURRED

Most SAS procedures require that a data set be specified after the PROC name. PROC DATASETS requires you to specify a library where your data set is located ①. The default behavior of the procedure is to produce in the LOG a list all data sets in that library. The NOLIST option suppresses that list ①. The MODIFY statement says that you want to make changes to a specific data set ②. The LABEL statement looks like the one you have seen in previous examples within data steps, however this time it is in a PROC ③. The RENAME statement changes the names to uppercase (old name = new name) ④. The procedure ends with a QUIT (not a RUN) statement ⑤. The addition of labels and renaming of variables could also have been accomplished in a data step, but that data step also would have read and written all the observations in the data set just to change some header information ... a less efficient approach ...

```
data x.cancer99;
set x.cancer99.
label <rest of label statement>;
rename <rest of rename statement>;
run;
```

If you want to alter a data set and add/remove labels, change variable names, or add/remove formats, use PROC DATASETS, not a data step. There are many other tasks you can accomplish with PROC DATASETS, but these are enough for you to know (many SAS users never use the procedure so you are already ahead of them knowing only this much).

Now that the data set has new variable names and labels, the last item is to show what all the values of the variables mean. There are too many county codes to list. The values of both age and cause are explained in the labels in example 6.4. That leaves two variables that have codes that need explanation, gender and place.

variable	values
gender	1=male, 2=female
place	A=hospital (DOA), B=hospital (ER), C=hospital (outpatient), D=hospital (inpatient), E=other institution, F=decedent's residence, G=other private home, H=other non-institution, N=not-in-hospital, Z=unknown (or not classifiable)

...PROC PRINT/REPORTING...

There are numerous PROCs that can be used to create reports in SAS. If a procedure cannot produce the type of output you want, you can use a data step to create a custom report. One of the easiest ways to create a report is with PROC PRINT. As stated in the SAS online help...

The PRINT procedure prints the observations in a SAS data set for simple reports. PROC PRINT prints a listing of the values of some or all of the variables in a SAS data set. You can produce customized reports using procedure options and statements.

The default behavior of PROC PRINT is to print all variables and all observations. If variables are formatted, the formatted values are printed ...

```
proc print data=x.cancer99;
run;
```

would produce a reports with 35,000+ lines, listing the values for the five variables in that data set. A portion of the data set CANCER99 can be printed by using the FIRSTOBS and OBS data set options discussed earlier in chapter 4.

...Example 6.5...

```
title 'A PORTION OF DATA SET CANCER99 (SELECTED BY OBSERVATION NUMBER)';
proc print data=x.cancer99 (firstobs=501 obs=510); ①
run;

proc print data=x.cancer99 (firstobs=501 obs=510) label obs='OBSERVATION NUMBER; ②
run;
```

Two data set options are used to limit the output of PROC PRINT to the ten observations from 501 through 510 ①. The output lists observation numbers in the first column, with the order of remaining columns determined by the order of the variables in the data set. Notice that the variable names are all in uppercase, the change made using PROC DATASETS in example 6.4. Also notice that the labels added in example 6.4 are not used.

A PORTION OF DATA SET CANCER99 (SELECTED BY OBSERVATION NUMBER)

Obs	COUNTY	GENDER	AGE	CAUSE	PLACE
501	33	2	79	C56	B
502	33	2	77	C509	D
503	33	1	72	C349	D
504	33	2	79	C183	D
505	33	1	52	C349	F
506	33	2	73	C259	F
507	05	1	76	C819	D
508	33	1	65	C349	D
509	33	1	80	C64	F
510	26	2	45	C449	D

The default behavior of PROC PRINT is to use variable names, not labels as column headings. The LABEL option puts labels rather than variable names as the column headers ②. Also, you can change the default text of 'Obs' over column one with the OBS option ②. The changes that result from using these two options are shown on the right.

A PORTION OF DATA SET CANCER99 (SELECTED BY OBSERVATION NUMBER)

OBSERVATION NUMBER	GAZETTEER COUNTY NUMBER	GENDER	AGE AT DEATH (YEARS)	ICD-10 CAUSE OF DEATH	WHERE DEATH OCCURRED
501	33	2	79	C56	B
502	33	2	77	C509	D
503	33	1	72	C349	D
504	33	2	79	C183	D
505	33	1	52	C349	F
506	33	2	73	C259	F
507	05	1	76	C819	D
508	33	1	65	C349	D
509	33	1	80	C64	F
510	26	2	45	C449	D

In addition to using data set option to limit the number of observations that are printed, you can also use a WHERE statement of data set option (as described in chapter 4). Rather than look at a pre-specified range of observations, you can use a random number function within a WHERE statement (more about functions in a later chapter).

...Example 6.6...

```
title 'A PORTION OF DATA SET CANCER99 (SELECTED RANDOMLY - REPEATABLE)';
proc print data=x.cancer99 label;
where ranuni(99) le .0005; ①
run;

title 'A PORTION OF DATA SET CANCER99 (SELECTED RANDOMLY - NON-REPEATABLE)';
proc print data=x.cancer99 label;
where ranuni(0) le .0005; ②
run;
```

A PORTION OF DATA SET CANCER99 (SELECTED RANDOMLY - REPEATABLE)

Obs	GAZETTEER COUNTY NUMBER	GENDER	AGE AT DEATH (YEARS)	ICD-10 CAUSE OF DEATH	WHERE DEATH OCCURRED
2689	29	1	68	C349	F
5275	32	2	83	C169	D
7125	27	1	62	C719	E
8069	56	1	67	C444	D
8244	29	1	73	C259	D
8703	31	2	58	C189	D
12635	51	2	82	C509	D
13034	46	1	74	C859	D
13200	96	1	30	C920	D
13782	50	1	90	C349	D
14069	51	2	76	C349	D
14811	14	2	70	C509	F
15195	07	2	79	C259	F
15584	94	2	81	C859	E
16897	51	2	83	C23	E
18805	51	2	64	C859	F
21178	33	1	72	C159	G
22745	12	1	85	C189	E
30288	96	1	84	C64	D
32608	94	2	65	C349	D
34633	94	2	57	C80	D
36574	95	2	78	C787	D

A PORTION OF DATA SET CANCER99 (SELECTED RANDOMLY - NON-REPEATABLE)

Obs	GAZETTEER COUNTY NUMBER	GENDER	AGE AT DEATH (YEARS)	ICD-10 CAUSE OF DEATH	WHERE DEATH OCCURRED
3447	54	2	66	C349	F
5382	45	2	39	C439	F
10985	41	2	97	C509	D
12121	29	2	51	C349	E
13913	29	1	82	C169	D
14212	59	1	89	C499	F
20919	41	2	72	C20	E
21508	09	1	57	C349	D
22866	33	1	92	C679	E
22938	14	1	93	C80	D
23495	96	1	62	C679	D
24235	95	1	80	C349	D
24710	95	2	67	C509	D
25287	96	2	92	C349	B
25545	93	2	76	C259	D
25625	96	2	66	C509	D
25853	96	1	53	C189	N
26311	94	2	73	C509	D
26745	93	2	83	C56	D
27857	93	2	48	C80	D
28119	96	1	84	C64	N
32932	59	2	64	C259	D

The output from PROC PRINT on the left is the result of using a WHERE statement that contains the RANUNI random number function ①. That function generates uniform (UNI) random (RAN) numbers in the range 0 to 1. If you ask SAS to use an observation each time it is less than .0005, approximately 0.05% of the observations are printed ... if you take 0.05% of the 35,000+ observations in data set CANCER99, you will get approximately 20 observations printed. The argument (the number within the parentheses following RANUNI) is a positive integer, 99, called a seed number. If you use the function again with that number, you will get the same observations printed. The output is repeatable since using the same seed number repeats the same set of random numbers.

The output on the right is also the result of using the RANUNI function, but it is not repeatable. A seed number of 0 is used ②. That seed number causes SAS to use the time produced by the system clock as the real seed number used to generate the random numbers, a random starting point. If you want your output to be repeatable, use a positive integer in the RANUNI function, otherwise use a zero.

All the options thus far have limited the observations that are printed. Since the default behavior is to print values of all variables, you should also know how to limit the variables that are printed.

...Example 6.7...

```
title1 'A PORTION OF DATA SET CANCER99 (SELECTED RANDOMLY - REPEATABLE)';
title2 'LIMIT VARIABLES WITH A PROC OPTION - THE VAR STATEMENT';
proc print data=x.cancer99 label;
var county cause; ①
where ranuni(99) le .0005;
run;

title1 'A PORTION OF DATA SET CANCER99 (SELECTED RANDOMLY - REPEATABLE)';
title2 'LIMIT VARIABLES WITH A DATA SET OPTION - DROP';
proc print data=x.cancer99 (drop=age) label;  ②
where ranuni(99) le .0005;
run;
```

Variables in PROC PRINT can be selected using a VAR statement ①. They can also be selected using a DROP (or KEEP) data set option ②. Your choice of method is dictated by the one that requires you to enter the shorter list of variables. In example 6.7, you are printing a data set that has five variables. If you only want two printed (COUNTY and CAUSE), listing two is shorter that listing three not to print in a DROP option. If you want four printed, it is easier to use a DROP option (do not print AGE).

There are a number of other options that you can use with PROC PRINT. For now, you can limit that list to: NOOBS, suppresses the observation number column; DOUBLE, produces double-spaced output. There are also two statements that are useful (in addition to VAR): SUM, produces totals for columns of one or more numeric variables; ID, repeats a variable at the beginning of a row of procedure output (useful when printing data sets with many variables that do not all fit on one row).

...Example 6.8...

```
data nyc;
input county : $10. population births deaths; ①
format population births deaths comma9.;
datalines;
BRONX 1233100 22784 10206
KINGS 2383700 39967 18412
NEW_YORK 1515300 19443 11594
QUEENS 2062300 31922 16219
RICHMOND 414800 5767 3428
;
run;

title 'EXAMPLE OF A SUM STATEMENT';
proc print data=nyc noobs double; ②
sum population births deaths; ③
run;
```

EXAMPLE OF A SUM STATEMENT

county	population	births	deaths
BRONX	1,233,100	22,784	10,206
KINGS	2,383,700	39,967	18,412
NEW_YORK	1,515,300	19,443	11,594
QUEENS	2,062,300	31,922	16,219
RICHMOND	414,800	5,767	3,428
	9	9	9
	7,609,200	119,883	59,859

A new data set is created containing three numeric variables ①. PROC PRINT is used to display the values of the variables. The NOOBS option suppresses observation numbers while the DOUBLE option results in double-spaced output ②. All three numeric variables are listed in a SUM statement ③. Notice that the variable COUNTY also appears. The default output without using a VAR statement is still to print all the variables. The SUM statement just produces column totals, it does not specify a list of variables to print.

...PROC FREQ/COUNTING...

There are a number of procedures that can produce counts. Just as PROC PRINT is perhaps the most elemental way to create reports, PROC FREQ is the basic counting procedure. PROC FREQ also has a number of options that allow you to compute statistics (chi-square, Mantel-Haenszel, etc.). As stated in the SAS online help ...

The FREQ procedure produces 1-way to n-way frequency and cross tabulation tables. Frequency tables show the distribution of variable values. Cross tabulation tables show combined frequency distributions for two or more variables. For one-way tables, PROC FREQ can compute chi-square tests for equal or specified proportions. For two-way tables, PROC FREQ computes tests and measures of association. For n-way tables, PROC FREQ does stratified analysis, computing statistics within as well as across strata.

The default behavior of PROC FREQ is to produce a table for all variables using all observations. If variables are formatted, the formatted values are used in the tables ...

```
proc print data=x.cancer99;
run;
```

would produce a five tables, one for each variable in the data set. There would be counts within each table of the number of observations having a specific value of the variable being used in that table. If you ran the above SAS job, some of the tables would be very small. There are only two values of the variable GENDER, so that table would only have two rows. However, the data set has over 200 different causes of deaths, so that table would have 200+ rows. The previous section on PROC PRINT started with how to restrict the number of observations used. This section starts with how to limit the number of variables used.

...Example 6.9...

```
title 'SPECIFYING TABLES USED IN PROC FREQ - SIMPLE TABLE STATEMENT' ;
proc freq data=x.cancer99;
table gender place; ①
run;

title 'SPECIFYING TABLES USED IN PROC FREQ - USE MISSING DATA' ;
proc freq data=x.cancer99;
table gender place / missing; ②
run;

title 'SPECIFYING TABLES USED IN PROC FREQ - INSERT MISSING DATA' ;
proc freq data=x.cancer99;
table gender place / missprint; ③
run;
```

The statement used in PROC FREQ to select variables is a TABLE statement ①. In example 6.9, two variables are specified, GENDER and PLACE. In the output on the right, you can see the values that each variable has in the data set, how many times that value occurs, plus a cumulative frequency and a couple of percentages. Note the text at the bottom of the output. The default behavior of PROC FREQ is to leave observations with missing data out of the table. There are 149 observations in the data set with no PLACE listed (notice that there are no missing values for GENDER).

SPECIFYING TABLES USED IN PROC FREQ - SIMPLE TABLE STATEMENT

gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	18542	49.34	18542	49.34
2	19041	50.66	37583	100.00

place	Frequency	Percent	Cumulative Frequency	Cumulative Percent
A	127	0.34	127	0.34
B	812	2.17	939	2.51
C	301	0.80	1240	3.31
D	20242	54.07	21482	57.39
E	4367	11.67	25849	69.05
F	8257	22.06	34106	91.11
G	606	1.62	34712	92.73
H	18	0.05	34730	92.78
N	2618	6.99	37348	99.77
Z	86	0.23	37434	100.00

Frequency Missing = 149

If you want to determine the percentage of observations that have a missing value for the variable PLACE, you would have to get out your calculator (or pencil and paper). The MISSING option in PROC FREQ adds missing values to the output and includes the values in counts and percentages ②. On the right, it is now easy to see that missing data comprise less than one percent of the values for the variable PLACE. Another option, MISSPRINT, places the missing data in the table, but does not include the values in computing either counts or percentages ③. The total number of observations with missing data is still added after the table when MISSPRINT is used. The usefulness of this option is more apparent when producing tables with more than one dimension.

There are many other options available in PROC FREQ. One them is used to suppress the printing of percentages in simple tables, while the other suppresses cumulative counts and percentages.

SPECIFYING TABLES USED IN PROC FREQ - USE MISSING DATA

place	Frequency	Percent	Cumulative Frequency	Cumulative Percent
	149	0.40	149	0.40
A	127	0.34	276	0.73
B	812	2.16	1088	2.89
C	301	0.80	1389	3.70
D	20242	53.86	21631	57.56
E	4367	11.62	25998	69.17
F	8257	21.97	34255	91.14
G	606	1.61	34861	92.76
H	18	0.05	34879	92.81
N	2618	6.97	37497	99.77
Z	86	0.23	37583	100.00

SPECIFYING TABLES USED IN PROC FREQ - INSERT MISSING DATA

place	Frequency	Percent	Cumulative Frequency	Cumulative Percent
	149	.	.	.
A	127	0.34	127	0.34
B	812	2.17	939	2.51
C	301	0.80	1240	3.31
D	20242	54.07	21482	57.39
E	4367	11.67	25849	69.05
F	8257	22.06	34106	91.11
G	606	1.62	34712	92.73
H	18	0.05	34730	92.78
N	2618	6.99	37348	99.77
Z	86	0.23	37434	100.00

Frequency Missing = 149

...Example 6.10...

```
title 'SUPPRESSING PERCENTGAES' ;
proc freq data=x.cancer99;
table gender / nopercent; ①
run;

title 'SUPPRESSING CUMULATIVE COUNTS AND PERCENTAGES' ;
proc freq data=x.cancer99;
table gender / nocum; ②
run;

title 'JUST COUNTS' ;
proc freq data=x.cancer99;
table gender / nopercent nocum; ③
run;
```

SUPPRESSING PERCENTGAES

gender	Frequency	Cumulative Frequency
1	18542	18542
2	19041	37583

SUPPRESSING CUMULATIVE COUNTS AND PERCENTAGES

gender	Frequency	Percent
1	18542	49.34
2	19041	50.66

JUST COUNTS

gender	Frequency
1	18542
2	19041

The NOPERCENT option eliminates both percentages ①. NOCUM eliminates all both the cumulative counts and percentages, but leaves the regular percentages ②. If you use both options, the table includes only counts ③.

When variables have a large number of different values, tables produced by PROC FREQ will be large. You can reduce the size of tables in a number of different ways. One is to use a WHERE statement to reduce the number of observations used to create a table. Another is to use formats to reduce the number of different categories of a variable that appear in a table. User-written formats are discussed in the next chapter. There are also SAS-supplied formats that can reduce table sizes.

...Example 6.11...

```
title 'REDUCED TABLE SIZE USING WHERE';
proc freq data=x.cancer99;
table age;
where age between 0 and 4; ①
run;

title 'REDUCED TABLE SIZE USING A FORMAT + (WHERE)';
proc freq data=x.cancer99;
table cause; ②
where age between 0 and 4;
format cause $3.; ③
run;
```

There are many individual ages in the data set. However, the WHERE statement limits the output to only those observations with an age in the range 0 to 4 (why not just ask for the observations with an age less than 5?) ①. There are also many causes, but the first three characters in the four character cause of death are enough to place a death in a major category. For example, all brain cancer causes of death begin with C71. You can create a table of causes of death ② using only the first three characters of the variable CAUSE by using a SAS-supplied format ③. You can see that in the counts on the right, all brain cancers are grouped into one entry in the table. Previous examples have use SAS-supplied numeric formats to reduce the number of decimal places displayed for variables created in a data step (for example a format of 4.1 to display age rounded to one decimal place). The SAS-supplied character format $3. asks that only the first three characters of the variable CAUSE be used in creating counts. The $3. may look familiar since you have seen similar looking informats used to read values of raw data using formatted input.

REDUCED TABLE SIZE USING WHERE

age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	7	19.44	7	19.44
1	6	16.67	13	36.11
2	8	22.22	21	58.33
3	6	16.67	27	75.00
4	9	25.00	36	100.00

REDUCED TABLE SIZE USING A FORMAT (+ WHERE)

cause	Frequency	Percent	Cumulative Frequency	Cumulative Percent
C41	1	2.78	1	2.78
C43	1	2.78	2	5.56
C49	2	5.56	4	11.11
C69	1	2.78	5	13.89
C71	11	30.56	16	44.44
C74	2	5.56	18	50.00
C79	2	5.56	20	55.56
C80	5	13.89	25	69.44
C85	1	2.78	26	72.22
C91	1	2.78	27	75.00
C92	6	16.67	33	91.67
C94	1	2.78	34	94.44
C95	1	2.78	35	97.22
C97	1	2.78	36	100.00

The default order of the display of counts in PROC FREQ is the order of the values of the observations. This is referred to as INTERNAL order. The order can be change using an ORDER option and one very useful value of the ORDER option is FREQ.

...Example 6.12...

```
title 'TABLE IN FREQUENCY ORDER';
proc freq data=x.cancer99 order=freq; ①
table cause;
where age between 45 and 64 and gender eq '2'; ②
format cause $3.; ③
run;
```

TABLE IN FREQUENCY ORDER

cause	Frequency	Percent	Cumulative Frequency	Cumulative Percent
C34	1118	24.42	1118	24.42
C50	996	21.76	2114	46.18
C18	307	6.71	2421	52.88
C56	297	6.49	2718	59.37
C25	241	5.26	2959	64.64
C80	186	4.06	3145	68.70
C53	141	3.08	3286	71.78
C85	123	2.69	3409	74.46
C71	105	2.29	3514	76.76
C16	83	1.81	3597	78.57
C55	74	1.62	3671	80.19
C92	64	1.40	3735	81.59
C90	62	1.35	3797	82.94
C54	60	1.31	3857	84.25
C22	56	1.22	3913	85.47
C15	53	1.16	3966	86.63
C64	51	1.11	4017	87.75

The ORDER=FREQ option changes the table order from INTERNAL to frequency order, with the highest counts first ①. A WHERE statement limits the observations used in the table to females (gender '2') in the age range 45 to 64 ②, and a format statement creates counts using only the first three characters of the four character cause of death ③. The output from PROC FREQ shows that lung cancer (C34) and breast cancer (C50) account for almost 50% of cancer deaths among females age 45 to 64 and that each of these causes accounts for more than 3x the number of deaths from any other cancer site.

Another ORDER option is FORMATTED. Examples 6.11 and 6.12 both used a SAS-supplied format to group observations. Though user-written formats are not discussed until chapter 7, the next example shows how a user-written format can group observations and order the output of PROC FREQ.

...Example 6.13...

```
proc format; ①
value $place
'A'-'C'     = 'HOSPITAL (OTHER)'
'D'         = 'HOSPITAL INPATIENT'
'E'         = 'OTHER INSTITUTION'
'F'         = "DECEDENT'S RESIDENCE"
'G','H','N' = 'OTHER NON-HOSPITAL'
'Z'         = 'UNKNOWN (OR NOT CLASSIFIABLE)';
run;

title 'TABLE IN FORMATTED ORDER';
proc freq data=x.cancer99 order=formatted;
table place;
format place $place.; ②
run;
```

PROC FORMAT is used to create the format name $PLACE ①. Values to the left of the ='s are those found in the data set, while those to the right of the ='s are labels that are to be used in lieu of data values. The format is used in PROC FREQ to create the table on the right ②. You can see that entries in the table are in alphabetic order according to the label create in PROC FORMAT. Also, observations are grouped based on the values you see in PROC FORMAT on the left of the ='s. Right now, the details of PROC FORMAT are not important. What is important is to remember that FORMATTED order is available in PROC FREQ using the ORDER option.

TABLE IN FORMATTED ORDER

place	Frequency	Percent	Cumulative Frequency	Cumulative Percent
DECEDENT'S RESIDENCE	8257	22.06	8257	22.06
HOSPITAL (OTHER)	1240	3.31	9497	25.37
HOSPITAL INPATIENT	20242	54.07	29739	79.44
OTHER INSTITUTION	4367	11.67	34106	91.11
OTHER NON-HOSPITAL	3242	8.66	37348	99.77
UNKNOWN (OR NOT CLASSIFIABLE)	86	0.23	37434	100.00

Frequency Missing = 149

A table with only one variable is referred to as having one dimension. There is no limit as to the number of dimensions in tables you can create with PROC FREQ. Dimensions are added to tables by placing an asterisk rather than a space between the names of variables in the TABLE statement.

...Example 6.14...

```
title 'A TWO-DIMENSION TABLE';
proc freq data=x.cancer99;
table gender*age; ①
where age between 0 and 4; ②
run;
```

The table statement contains the name of two variables (GENDER and AGE) separated by an asterisk ①. The table on the right shows the two-dimension table, restricted to deaths in the age range 0 to 4 using a WHERE statement ②. You should take note of a number of features of this table: the variable used last in the table statement is a column variable, while the variable used first is a row variable; in addition to counts, there are percentages; a box in the upper left contains a description of the contents of the table cells; there are marginal counts and percentages; there is an overall count. There are options for the TABLE statement that will suppress one or more of the percentages.

A TWO-DIMENSION TABLE

Table of gender by age

gender age

Frequency Percent Row Pct Col Pct	0	1	2	3	4	Total
1	3 8.33 15.79 42.86	2 5.56 10.53 33.33	6 16.67 31.58 75.00	4 11.11 21.05 66.67	4 11.11 21.05 44.44	19 52.78
2	4 11.11 23.53 57.14	4 11.11 23.53 66.67	2 5.56 11.76 25.00	2 5.56 11.76 33.33	5 13.89 29.41 55.56	17 47.22
Total	7 19.44	6 16.67	8 22.22	6 16.67	9 25.00	36 100.00

...Example 6.15...

```
title 'SUPPRESSING PERCENTAGES IN AA TWO-DIMENSION TABLE';
proc freq data=x.cancer99;
table gender*age / norow; ①
table gender*age / nocol; ②
table gender*age / nopercent; ③
table gender*age / norow nocol nopercent; ④
where age between 0 and 4;
run;
```

SUPPRESSING PERCENTAGES IN A TWO-DIMENSION TABLE

Table of gender by age

gender age

Frequency Percent Col Pct	0	1	2	3	4	Total
1	3 8.33 42.86	2 5.56 33.33	6 16.67 75.00	4 11.11 66.67	4 11.11 44.44	19 52.78
2	4 11.11 57.14	4 11.11 66.67	2 5.56 25.00	2 5.56 33.33	5 13.89 55.56	17 47.22
Total	7 19.44	6 16.67	8 22.22	6 16.67	9 25.00	36 100.00

Table of gender by age

gender age

Frequency Percent Row Pct	0	1	2	3	4	Total
1	3 8.33 15.79	2 5.56 10.53	6 16.67 31.58	4 11.11 21.05	4 11.11 21.05	19 52.78
2	4 11.11 23.53	4 11.11 23.53	2 5.56 11.76	2 5.56 11.76	5 13.89 29.41	17 47.22
Total	7 19.44	6 16.67	8 22.22	6 16.67	9 25.00	36 100.00

SUPPRESSING PERCENTAGES IN A TWO-DIMENSION TABLE

Table of gender by age

gender age

Frequency Row Pct Col Pct	0	1	2	3	4	Total
1	3 15.79 42.86	2 10.53 33.33	6 31.58 75.00	4 21.05 66.67	4 21.05 44.44	19
2	4 23.53 57.14	4 23.53 66.67	2 11.76 25.00	2 11.76 33.33	5 29.41 55.56	17
Total	7	6	8	6	9	36

Table of gender by age

gender age

Frequency	0	1	2	3	4	Total
1	3	2	6	4	4	19
2	4	4	2	2	5	17
Total	7	6	8	6	9	36

There are three options that suppress percentages in a two-dimension table. The NOROW option eliminates row percentages ①, while the NOCOL option eliminates column percentages ②. In either case, the output on the bottom left of the previous page shows that the overall percentages and marginal percentages remain in the tables. If you look at the output on the bottom right of the previous, the upper table shows that the NOPERCENT option eliminates the overall percentages, but also removes the marginal percentages ③. You can use one or more of the above options in a TABLE statement. Using all three results in a table with only counts (lower right on previous page) ④.

Example 6.15 shows that you can have more than one TABLE statement in PROC FREQ, each with different options. If you want to produce more than one table, all having the same options, you can use one TABLE statement (also see example 6.9).

...Example 6.16...

```
proc format; ①
value $cause
'C15' - 'C159' = 'ESOPHA -GUS'
'C18' - 'C189' = 'COLON'
'C25' - 'C259' = 'PANCREAS'
'C34' - 'C349' = 'LUNG'
'C50' - 'C509' = 'BREAST'
'C56'          = 'OVARY'
other          = 'OTHER';
run;

title 'MULTIPLE TABLES WITH SAME OPTIONS';
proc freq data=x.cancer99;
table gender*cause gender*age / norow nocol nopercent; ②
where age between 45 and 54; ③
format cause $cause.; ④
run;
```

MULTIPLE TABLES WITH SAME OPTIONS

Table of gender by cause

gender cause

Frequency	OTHER	ESOPHA-GUS	COLON	PANCREAS	LUNG	BREAST	OVARY	Total
1	841	72	103	121	458	5	0	1600
2	615	17	109	60	335	439	120	1695
Total	1456	89	212	181	793	444	120	3295

Table of gender by age

gender age

Frequency	45	46	47	48	49	50	51	52	53	54	Total
1	98	112	123	142	143	187	176	210	200	209	1600
2	123	119	129	146	131	177	196	230	216	228	1695
Total	221	231	252	288	274	364	372	440	416	437	3295

Once again, PROC FORMAT (more in chapter 7) is used to create a rule to group observations, this time based on the cause of death ①. The TABLE statement in PROC FREQ requests two tables, both with all percentages suppressed ②. The output is limited to observations in the age range 45 to 54 ③, and the format created earlier is used to group observations based in the values of the variable CAUSE ④. The five causes used in PROC FREQ account for approximately 50% of male deaths and over 60% of female deaths in the age range 45 to 54.

In example 6.9, the MISSING and MISSPRINT options were used with a one-dimension table. The information provided with the MISSPRINT option seemed redundant in that the number of observations with missing data was listed in both the table and beneath the table. The MISSING and MISSPRINT options also work in two (or more)-dimension tables.

...Example 6.17...

```
proc format; ①
value $place
'A'-'C'     = 'HOSPITAL (OTHER)'      'D'         = 'HOSPITAL INPT'
'E'         = 'OTHER INST'            'F'         = "DECEDENT'S RES"
'G','H','N' = 'OTHER NON-HOSP'        'Z'         = 'UNKNOWN';
value age
 0-4    = '<5'          5-14   = '5-14'
15-44   = '15-44'      45-64   = '45-64'
65-84   = '65-84'      85-high = '85+';
run;

title 'MISSPRINT OPTION IN A TWO-DIMENSION TABLE';
proc freq data=x.cancer99;
table place * age / missprint norow nocol; ②
format place $place. age age.;
run;
```

PROC FORMAT creates two formats that are to be used in PROC FREQ to group observations (more in chapter 7) ①. The MISSPRINT option is used on the TABLE statement to place counts of missing data in the body of the table ②. Without that option, all you would know about the missing data is that there are 154 observations with missing data for either AGE or PLACE. You would not know how many observations have missing data for both variables (you can see that count is 0 in the cell on the upper left of the table). The count of 154 at the bottom of the table is not redundant as it was in the one-dimension table example since without it, you would have to add up all the counts of missing data in the table.

MISSPRINT OPTION IN A TWO-DIMENSION TABLE

Table of PLACE by AGE

PLACE(WHERE DEATH OCCURRED) AGE(AGE AT DEATH (YEARS))

Frequency Percent	.	<5	5-14	15-44	45-64	65-84	85+	Total
	0 .	0 .	0 .	7 .	40 .	76 .	26 .	. .
HOSPITAL (OTHER)	1 .	0 0.00	3 0.01	66 0.18	357 0.95	669 1.79	144 0.38	1239 3.31
HOSPITAL INPT	2 .	25 0.07	49 0.13	1016 2.71	5547 14.82	11205 29.94	2398 6.41	20240 54.08
OTHER INST	0 .	0 0.00	0 0.00	61 0.16	559 1.49	2493 6.66	1254 3.35	4367 11.67
DECEDENT'S RES	2 .	9 0.02	9 0.02	260 0.69	2081 5.56	4913 13.13	983 2.63	8255 22.06
OTHER NON-HOSP	0 .	2 0.01	4 0.01	93 0.25	720 1.92	1872 5.00	551 1.47	3242 8.66
UNKNOWN	0 .	0 0.00	0 0.00	5 0.01	15 0.04	51 0.14	15 0.04	86 0.23
Total	. .	36 0.10	65 0.17	1501 4.01	9279 24.79	21203 56.65	5345 14.28	37429 100.00

Frequency Missing = 154

There are a few items to notice in the above table: the row label space (with formatted variable values of PLACE on the left) is 16 spaces wide and any text longer than that wraps at 16 characters; the column label space (with formatted variable values of AGE on the top) is 8 spaces wide and any text longer than that wraps at 8 characters; the table was constructed using the CANCER99 data set modified with PROC DATASETS in example 6.4 so in addition to variable names, you also see variable labels (and uppercase variable names).

The description of PROC FREQ in the SAS online help says that it can create 1-way to n-way frequency and cross tabulation tables. With a two-dimension table, you have seen that the last variable in the

TABLE statement is the columns variable, with the first variable being the row variable. What happens when a third variable is added to the TABLE statement.

...Example 6.18...

```
title 'THREE-DIMENSION TABLE';
proc freq data=x.cancer99;
table gender*place*age / norow nocol nopercent; ①
where age between 0 and 4; ②
run;
```

Three variables are used in the TABLE statement ① and a WHERE statement limits the counts to only those in the age range 0 to 4. The last variable, AGE, in the TABLE statement is a column variable just as when creating a 2-dimension table. The next-to-last variable, PLACE, is a row variable just as when creating a 2-dimension table. A two-dimension table is created for each value if the third variable, GENDER. Notice that the table on the top right has some entries with zero marginal total, places G and N. Normally, PROC FREQ does not count what is not present in your data. However, when a third variable is added, all tables for the various values of the third variable will have the same number of rows and columns, even if that results in zero counts. If a fourth variable is added to the TABLE statement, a two-dimension table is created for every combination of values for the 3rd and 4th variables. You can see that if you are using variables with many different values, there will be a lot of 2-dimension tables. For example, since there are 2 values of the variable GENDER and 62 values of the variable county, a table statement including COUNTY*GENDER*PLACE*AGE results in 124 tables. There is another way to produce multiple 2-dimension tables and that is described in by-group processing in a later section of this chapter.

THREE-DIMENSION TABLE

Table 1 of PLACE by AGE
Controlling for GENDER=1

PLACE(WHERE DEATH OCCURRED) AGE(AGE AT DEATH (YEARS))

Frequency	0	1	2	3	4	Total
D	3	1	3	1	3	11
F	0	1	3	3	1	8
G	0	0	0	0	0	0
N	0	0	0	0	0	0
Total	3	2	6	4	4	19

Table 2 of PLACE by AGE
Controlling for GENDER=2

PLACE(WHERE DEATH OCCURRED) AGE(AGE AT DEATH (YEARS))

Frequency	0	1	2	3	4	Total
D	4	3	2	1	4	14
F	0	0	0	1	0	1
G	0	1	0	0	0	1
N	0	0	0	0	1	1
Total	4	4	2	2	5	17

If you want to produce a number of 2-dimension tables, all of which contain the same variable, there is a shortcut you can use in the TABLE statement.

...Example 6.19...

```
title 'MODIFIED TABLE STATEMENT';
proc freq data=x.cancer99;
table gender*(place age) / norow nocol nopercent; ①
where age between 0 and 4;
run;
```

MODIFIED TABLE STATEMENT

Table of GENDER by PLACE

GENDER PLACE(WHERE DEATH OCCURRED)

Frequency	D	F	G	N	Total
1	11	8	0	0	19
2	14	1	1	1	17
Total	25	9	1	1	36

Table of GENDER by AGE

GENDER AGE(AGE AT DEATH (YEARS))

Frequency	0	1	2	3	4	Total
1	3	2	6	4	4	19
2	4	4	2	2	5	17
Total	7	6	8	6	9	36

The TABLE statement contains three variables as in the last example ①. However, written in this manner, it requests 2-dimension tables with GENDER in each table. One table is created with counts by GENDER and PLACE, another contains counts by GENDER and AGE.

Some of the statistics-related uses of PROC FREQ are discussed in a later section of this chapter.

...PROC MEANS/STATISTICS...

PROC MEANS is just one of a number or procedures that computes descriptive statistics for numeric variables in a SAS data set. It is also the easiest way to compute decriptive statistics. As stated in SAS online help ...

PROC MEANS computes statistics for an entire SAS data set or for groups of observations in the data set. If you use a BY statement, PROC MEANS calculates descriptive statistics separately for groups of observations. Each group is composed of observations having the same values of the variables used in the BY statement. The groups can be further subdivided by the use of the CLASS statement. PROC MEANS can optionally create one or more SAS data sets containing the statistics calculated. PROC MEANS is the easiest and most direct descriptive procedure for computing univariate statistics.

The default behavior of PROC MEANS is to produce descriptive statistics for all numeric variables using all observations with non-missing values. The default statistics are the mean, standard deviation, minimum, and maximum.

...Example 6.20...

```
title 'DEFAULT OUTPUT OF PROC MEANS';
proc means data=x.cancer99;
run;
```

There is only one numeric variable in the data set CANCER99, AGE. There are 37,583 observations in the data set, but you can see on the right that 37,578 are used to compute the various statistics since 5 observations have a missing value for AGE. You should also notice that the procedure provides more decimal places than the data deserve (if you go by the rule "*report only one more decimal place in the summary statistics than are present in your data*"). If you have multiple numeric variables in a data set, you can use a VAR statement to specify variables for analysis.

DEFAULT OUTPUT OF PROC MEANS

Analysis Variable : AGE AGE AT DEATH (YEARS)

N	Mean	Std Dev	Minimum	Maximum
37578	70.5725691	13.8227372	0	112.0000000

...Example 6.21...

```
data nyc;
input county : $10. population births deaths; ①
format population births deaths comma9.;
datalines; ②
BRONX 1233100 22784 10206
KINGS 2383700 39967 18412
NEW_YORK 1515300 19443 11594
QUEENS 2062300 31922 16219
RICHMOND 414800 . 3428
;
run;

title 'SELECT VARIABLES FOR ANALYSIS WITH A VAR STATEMENT';
proc means data=nyc maxdec=1; ③
var births deaths; ④
run;
```

SELECT VARIABLES FOR ANALYSIS WITH A VAR STATEMENT

Variable	N	Mean	Std Dev	Minimum	Maximum
births	4	28529.0	9271.8	19443.0	39967.0
deaths	5	11971.8	5826.1	3428.0	18412.0

The data step creates a data set with three numeric variables ①. Notice that there will be a missing value for BIRTHS in the last observation ②. A MAXDEC option is used in PROC MEANS to limit the number of decimal places that are reported ③ and a VAR statement specifies the variables to be analyzed ④. You should notice that all five observations are used to calculate statistics for DEATHS, but only the four observations with non-missing data for BIRTHS are used. You can also use a DROP data set option and leave out the VAR statement you want to analyze a large number of variables and leave out only a few.

In addition to the default statistics, there are a number of additional statistics that can be requested by adding statistics keywords as options. Some of the keywords in the following lists are obvious while some you will have to look up in online help (who remembers the definition of KURTOSIS).

Descriptive statistic keywords: CLM RANGE CSS SKEWNESS|SKEW CV STDDEV|STD KURTOSIS|KURT STDERR LCLM SUM MAX SUMWGT MEAN UCLM MIN USS N VAR NMISS

Quantile statistic keywords: MEDIAN|P50 Q3|P75 P1 P90 P5 P95 P10 P99 Q1|P25 QRANGE

Hypothesis testing keywords: PROBT T

When you requests statistics using any of the above keywords, you override the default output. The statistics you request are the only ones produced, they are not computed in addition to the default output.

...Example 6.22...

```
title 'SPECIFYING STATISTICS KEYWORDS';
proc means data=x.cancer99 maxdec=1 mean lclm uclm min q1 median q3 max; ①
run;
```

SPECIFYING STATISTICS KEYWORDS

Analysis Variable : AGE AGE AT DEATH (YEARS)

Mean	Lower 95% CL for Mean	Upper 95% CL for Mean	Minimum	Lower Quartile	Median	Upper Quartile	Maximum
70.6	70.4	70.7	0.0	62.0	73.0	80.0	112.0

A number of statistics keywords are used as options (in addition to the MAXDEC option) ①. The above output shows that PROC MEANS provides more descriptive labels than just repeating the keyword options. No VAR statement is used and the only numeric variable in the data set, AGE, is analyzed. All the non-missing values of AGE are used to compute statistics. You can also specify that you want your analyses done with values of another variable, for example, compute all statistics for males and females separately.

...Example 6.23...

```
title 'ANALYSIS WITH GROUPS USING A CLASS STATEMENT';
proc means data=x.cancer99 maxdec=1 mean lclm uclm p10 p25 p50 p75 p90; ①
class gender;  ②
run;
```

ANALYSIS WITH GROUPS USING A CLASS STATEMENT

Analysis Variable : AGE AGE AT DEATH (YEARS)

GENDER	N Obs	Mean	Lower 95% CL for Mean	Upper 95% CL for Mean	10th Pctl	25th Pctl	50th Pctl	75th Pctl	90th Pctl
1	18542	70.1	69.9	70.3	52.0	62.0	72.0	79.0	85.0
2	19041	71.0	70.8	71.2	52.0	63.0	73.0	81.0	87.0

A different set of statistics keywords is used (notice the different labels for the same values in the output, the MEDIAN is now 50TH PCTL) ①. The CLASS statement specifies that statistics should be computed for each value of the variable GENDER ②.

Just as you can specify multiple variables in a VAR statement, you can use more that one variable in a CLASS statement. Statistics will be calculated for all combinations of the values of all class variables.

...Example 6.24...

```
title 'ANALYSIS WITH GROUPS USING CLASS STATEMENT WITH TWO VARIABLES AND A SAS-SUPPLIED FORMAT';
proc means data=x.cancer99 maxdec=1 mean lclm uclm p10 p25 p50 p75 p90;
class cause gender; ①7
where cause in : ('C18' 'C34' 'C50' 'C56' 'C61' 'C71' 'C81'); ②
format cause $3.; ③
run;
```

ANALYSIS WITH GROUPS USING CLASS STATEMENT WITH TWO VARIABLES AND A SAS-SUPPLIED FORMAT

Analysis Variable : AGE AGE AT DEATH (YEARS)

ICD-10 CAUSE OF DEATH	GENDER	N Obs	Mean	Lower 95% CL for Mean	Upper 95% CL for Mean	10th Pctl	25th Pctl	50th Pctl	75th Pctl	90th Pctl
C18	1	1705	72.5	72.0	73.1	56.0	65.0	74.0	81.0	87.0
	2	1940	75.2	74.7	75.8	57.0	68.0	77.0	85.0	90.0
C34	1	5317	69.4	69.1	69.7	54.0	62.0	71.0	77.0	83.0
	2	4319	70.5	70.1	70.8	54.0	63.0	72.0	79.0	85.0
C50	1	41	70.7	66.1	75.3	51.0	60.0	74.0	80.0	87.0
	2	3113	68.4	67.9	68.9	47.0	57.0	70.0	81.0	87.0
C56	2	979	68.7	67.9	69.6	50.0	59.0	70.0	79.0	85.0
C61	1	2050	78.1	77.7	78.5	66.0	72.0	79.0	85.0	90.0
C71	1	409	60.0	58.3	61.7	39.0	50.0	62.0	73.0	79.0
	2	340	62.9	60.8	64.9	38.5	52.0	67.0	76.0	84.0
C81	1	50	53.8	48.7	59.0	29.5	37.0	55.5	70.0	75.5
	2	51	58.5	52.7	64.3	27.0	39.0	65.0	77.0	84.0

Two variables are used in the CLASS statement, CAUSE and GENDER ①. That order will be used in the output from PROC MEANS. Analysis is restricted to a group of causes ②. Notice that a colon (:) precedes the list of causes inside the parentheses. You are requesting that you want to use all causes that start with the specified characters, regardless of the 4th character of the cause. Since all causes have four characters, you use a SAS-supplied format to group observations based on the first three characters of the cause ③. The output lists the causes in the first column on the left since CAUSE is the first variable in the CLASS statement. Values of the variable GENDER are shown within each cause, as are the requested statistics. There is another way to produce analysis in groups other than the CLASS statement and that is described in by-group processing in a later section of this chapter.

In case you are curious, the various cancer causes are: C18, colon; C34, lung; C50, breast (yes, a small number of men die of breast cancer); C56, ovary; C61, prostate; C71, brain; C81, Hodgkin's Disease.

What you should remember...
For PROCs PRINT, FREQ, and MEANS you should know what default output is created when you specify the minimum amount of information (and what the minimum information is). Also, you should know how to select both variables and observations to be used the those PROCs. For each PROC, you should know the common options. You should know how to create 1,2, and 3 dimension tables in PROC FREQ. You should know how to produce analysis in groups in PROC MEANS. Finally, you should know how to use PROC CONTENTS to look at the attributes of your data and PROC DATASETS to modify some of those attributes.

...PROC SORT/REARRANGING OBSERVATIONS...

PROC SORT rearranges the order of observations in a SAS data set according to the values of one or more variables. As stated in SAS online help ...

The SORT procedure orders SAS data set observations by the values of one or more character or numeric variables. The SORT procedure either replaces the original data set or creates a new data set. PROC SORT produces only an output data set.

The default behavior of PROC SORT is to rearrange observations in ascending order. In addition to supplying the name of a data set to sort, PROC SORT also requires you to specify at least one variable whose values are used to sort observations. Data are sorted in most instances using ASCII order (see appendix B).

...Example 6.25...

```
proc sort data=x.cancer99;
by age; ①
run;

title 'DATA SET CANCER99 SORTED IN AGE ORDER';
proc print data=x.cancer99 (obs=15);
run;

proc sort data=x.cancer99;
by descending age; ②
run;

title 'DATA SET CANCER99 SORTED IN DESCENDING AGE ORDER';
proc print data=x.cancer99 (obs=15);
run;
```

DATA SET CANCER99 SORTED IN AGE ORDER

Obs	COUNTY	GENDER	AGE	CAUSE	PLACE
1	51	2	.	C259	F
2	59	2	.	C23	F
3	14	2	.	C679	D
4	14	1	.	C069	D
5	51	2	.	C439	B
6	33	1	0	C798	D
7	47	2	0	C959	D
8	95	1	0	C439	D
9	95	2	0	C719	D
10	96	2	0	C80	D
11	96	1	0	C80	D
12	29	2	0	C719	D
13	36	2	1	C910	D
14	57	2	1	C97	G
15	07	1	1	C859	F

DATA SET CANCER99 SORTED IN DESCENDING AGE ORDER

Obs	COUNTY	GENDER	AGE	CAUSE	PLACE
1	93	1	112	C349	C
2	51	2	108	C189	E
3	93	2	105	C189	D
4	51	1	103	C760	E
5	96	2	103	C169	B
6	93	1	103	C189	N
7	96	1	103	C189	D
8	94	1	103	C169	D
9	96	2	103	C189	D
10	06	1	102	C61	D
11	27	1	102	C329	E
12	29	2	102	C189	A
13	03	2	102	C509	F
14	51	2	102	C169	E
15	59	1	102	C61	D

The minimum amount of information is provided in PROC SORT, a BY statement specifying one variable, AGE ①. The output on the above right shows that the data are sorted in ascending order by age. Notice what values are listed first, the observations with missing values for AGE. They are considered as the lowest possible numeric (and character) value. When the DESCENDING option is used ②, the data are in descending order by age, output on lower right.

Sort Information

Sortedby	AGE
Validated	YES
Character Set	ANSI

If you run PROC CONTENTS after sorting a data set by one or more variables, you will notice some new information after the list of variables, shown here on the right. The header of the data set now contains the information that your data set is sorted in ascending order. If you had sorted your data set in descending order, that information is added to the data set header and you will see the word descending added in PROC CONTENTS. Since SAS can now easily determine if your data are sorted, if you try to sort and already sorted data set using the same by variable(s), you will see a message in the LOG ... shown here on the right. PROC SORT will not sort an already sorted data set unless the sort you request will change the already existing order of the observations.

Sort Information

Sortedby	DESCENDING AGE
Validated	YES
Character Set	ANSI

```
574  proc sort data=x.cancer99;
575  by age;
576  run;

NOTE: Input data set is already sorted, no sorting done.
```

Data sets can be sorted by more than one variable at a time. There is no limit as to the number of variables you can put in a BY statement ibn PROC SORT.

...Example 6.26...

```
proc sort data=x.cancer99;
by gender age; ①
run;

title 'DATA SET CANCER99 SORTED IN GENDER/AGE ORDER';
proc print data=x.cancer99 (obs=10);
run;

proc sort data=x.cancer99;
by gender descending age; ②
run;

title 'DATA SET CANCER99 SORTED IN GENDER/DESCENDING AGE ORDER';
proc print data=x.cancer99 (obs=10);
run;
```

If two variables are specified in a BY statement, the observations are sorted by the first variable, then by the second variable within values of the first variable ①. In the output on the above right, males (GENDER 1) are listed prior to females. Observations are sorted by AGE first within males, then within females. You can also use the DESCENDING option ②. That option only affects the variable immediately following the option. The output on the lower right still shows males first, but you see the older ages listed first in descending order. If you want the data sorted in descending order by both gender and age, you must repeat the option ...

DATA SET CANCER99 SORTED IN GENDER/AGE ORDER

Obs	COUNTY	GENDER	AGE	CAUSE	PLACE
1	14	1	.	C069	D
2	33	1	0	C798	D
3	95	1	0	C439	D
4	96	1	0	C80	D
5	07	1	1	C859	F
6	94	1	1	C80	D
7	46	1	2	C719	F
8	96	1	2	C920	D
9	45	1	2	C719	F
10	25	1	2	C499	F

DATA SET CANCER99 SORTED IN GENDER/DESCENDING AGE ORDER

Obs	COUNTY	GENDER	AGE	CAUSE	PLACE
1	93	1	112	C349	C
2	51	1	103	C760	E
3	93	1	103	C189	N
4	96	1	103	C189	D
5	94	1	103	C169	D
6	06	1	102	C61	D
7	27	1	102	C329	E
8	59	1	102	C61	D
9	14	1	101	C180	D
10	43	1	101	C349	F

```
proc sort data=x.cancer99;
by descending gender descending age;
run;
```

The fact that your data are now sorted by two variables is also added to the data set header and when PROC CONTENTS is run, you will see all the variables listed that were in the BY statement when the data set was sorted. You will also see the DESCENDING option listed in PROC CONTENTS.

In addition to sorting your data, PROC SORT can leave your original data set unsorted and produce a new copy of your data in sorted order. To do that, you must use the OUT option.

...Example 6.27...

```
proc sort data=x.cancer99 out=cause99; ①
by cause;
run;
```

A new data set, sorted by the variable CAUSE, is created in the WORK library ①. The new data set, CAUSE99, has all the variables and observations that are in data set CANCER99 and the only difference between the two data sets is that CAUSE99 is sorted by CAUSE.

When you sort data and create a new data set, you may not want to keep all the variables and/or observations from the original data set in the new data set. You can use DROP/KEEP data set options (not statements, they belong in data steps) to reduce the number of variables in the new data set. You can use a WHERE statement to select observations.

...Example 6.28...

```
proc sort data=x.cancer99 (drop=county place) out=cause99; ①
by cause;
where age ge 100; ②
run;
```

A new data set is created that will not include the variables COUNTY and PLACE ①. The DROP data set option is used on the original, not the new, data set. This is a more efficient way to create the new data set. If the option had been placed on the new data set, CAUSE99, you would sort observations with all variables, then write a reduced set of variables to the new data set. Since the DROP data set option is used on the data sort being sorted, observations with the reduced set of variables are first sorted, then written to the data set. The WHERE statement limits observations in the new data set to only those with an age of 100+ ②.

Remember that there is less work to do with the DROP option on the original rather than the new data set since shorter observations are being sorted. ***It takes approximately three times the size of the original data set to sort all the observations if you keep all the variables***. Anything you can do to reduce the size and number of observations being sorted makes the sorting progress run faster and use less disk space.

Two options available in PROC SORT allow you to get rid of duplicate observations. To see how these options work, two new small data sets are created. The first contains observations for several individuals who are seen at a clinic on multiple days. The second contains some mistakes, duplicate records that should not be in the data set. For the first data set, the task is to create a new data set with only one observation per individual, the first time they were seen at the clinic. For the second data set, the task is to get rid of any duplicate observations.

...Example 6.29...

```
data visits; ①
input id : $2 dov : mmddyy. group : $1. chol sbp dbp hr type : $1.;
format dov mmddyy10.;
datalines;
01 01/05/89 D 400 160  90 88 Y
01 02/15/89 D 350 156  88 80 Y
03 01/01/90 P 387 190 110 90 N
07 01/05/90 P 376 118  68 54 Y
05 01/06/90 P 399 188 110 92 N
03 02/13/90 P 377 188  96 84 Y
02 02/19/90 D 390 180 100 82 N
02 02/22/90 D 320 178  88 86 Y
02 02/25/90 D 325 172  82 78 Y
05 03/06/90 P 377 182 100 88 N
07 04/05/90 P 379 124  72 70 N
07 04/07/90 P 389 120  68 62 Y
;
run;

proc sort data=visits;
by id visit; ②
run;

proc sort data=visits out=firstvis nodupkey; ③
by id;
run;
```

```
title 'FIRST VISIT FOR EACH CLIENT';
proc print data=firstvis noobs;
run;
```

FIRST VISIT FOR EACH CLIENT

id	dov	group	chol	sbp	dbp	hr	type
1	01/05/1989	D	400	160	90	88	Y
2	02/19/1990	D	390	180	100	82	N
3	01/01/1990	P	387	190	110	90	N
5	01/06/1990	P	399	188	110	92	N
7	01/05/1990	P	376	118	68	54	Y

The data set VISITS is created and it contains multiple observations for clients with the same id number (duplicate values of the variable ID in the data set) ①. In addition to ID, the data set contains a visit date and a number of other variables. PROC SORT is used twice. The first time, the data set VISITS is sorted by ID and by VISIT with each client ②. After this step, the data set is in chronological order for each client. The second time the data set is sorted, a new data set name FIRSTVIS is created ③. Only ID is used in the BY statement. The NODUPKEY option eliminates any observations with duplicate values of the variable(s) in the BY statement, keeping only the first observation. Since the data are in date order for each client, the new data set contains only the observation with the earliest date of visit. If you want to create a data set with only the last visit for each client, you can add the DESCENDING option to the first sort ...

```
proc sort data=visits;
by id descending visit;
run;
```

Then, when duplicate observations are eliminated in the second sort, only the observation with the latest visit date will be kept.

...Example 6.30...

```
data duplicate; ①
input id gender age zip;
datalines;
1 1 15 12203
2 2 34 13502
1 1 15 12203
3 1 65 12201
4 2 34 12205
2 2 34 13502
1 1 15 12203
;
run;

proc sort data=duplicate out=single noduprecs; ②
by _all_;
run;

title 'DATA SET WITH DUPLICATE RECORDS ELIMINATED';
proc print data=single;
run;
```

The data set DUPLICATE is created and IDs 1 and 3 have duplicate observations, repeated observations where the values for all variables are the same. A new data set is created using PROC SORT ②. Data set DUPLICATE is sorted by all the variables using the shortcut _ALL_ a substitute for entering the names of all the variables in BY statement. The NODUPRECS option eliminates any observations that are exact matches of the previous observation. Since the data are sorted by all variables, all duplicate observations are eliminated.

DATA SET WITH DUPLICATE RECORDS ELIMINATED

Obs	id	gender	age	zip
1	1	1	15	12203
2	2	2	34	13502
3	3	1	65	12201
4	4	2	34	12205

...BY-GROUP PROCESSING...

Once data are sorted, you can use by-group processing (BGP) in a number of SAS procedures. BGP allows you create procedure output in a different manner than if your data were not sorted.

...Example 6.31...

```
proc sort data=x.cancer99 out=young; ①
by gender age;
where cause eq : 'C71' and age between 0 and 4;
run;

title 'BRAIN CANCER DEATHS, AGE < 5';
proc print data=young;
by gender; ②
run;
```

The data set YOUNG is created using PROC SORT ①. A WHERE statement limits the observations in the new data set to only those in the age range 0 to 4 and who died of brain cancer, causes starting with C74. The new data set is in order by gender and age within gender. A BY statement is used in PROC PRINT ②. The output on the right shows how the by variable GENDER is used. It is not a column in the output, rather its value is printed above sections of output.

BRAIN CANCER DEATHS, AGE < 5

GENDER=1

COUNTY	AGE	CAUSE	PLACE
46	2	C719	F
45	2	C719	F
93	2	C719	D
43	3	C716	F
31	4	C717	F

GENDER=2

COUNTY	AGE	CAUSE	PLACE
95	0	C719	D
29	0	C719	D
59	2	C719	D
51	3	C717	D
96	4	C717	N
29	4	C719	D

BGP in PROC FREQ can be used to reduce the number of dimensions in a table. If you wanted to use the data in data set YOUNG to create a table showing counts of the number of observations within categories of the variables GENDER and AGE, you create a 2-dimension table. You can also create the same count (and different looking output) using GENDER in a BY rather than TABLE statement.

...Example 6.32...

```
title 'BRAIN CANCER DEATHS, AGE < 5';
proc freq data=young;
table age;
by gender; ①
run;
```

The data set YOUNG is assumed to be still sorted by values of GENDER. GENDER is used in a BY statement, not in the TABLE statement ①. The counts in the table on the right could be found in a 2-dimension table, they are just presented differently with one variable in a BY statement and the other in a TABLE statement. Note that if you request multiple tables in one or more TABLE statements, all the tables among the males (GENDER 1) would be created first, then all tables among females (GENDER 2).

BRAIN CANCER DEATHS, AGE < 5

GENDER=1

AGE AT DEATH (YEARS)

AGE	Frequency	Percent	Cumulative Frequency	Cumulative Percent
2	3	60.00	3	60.00
3	1	20.00	4	80.00
4	1	20.00	5	100.00

BRAIN CANCER DEATHS, AGE < 5

GENDER=2

AGE AT DEATH (YEARS)

AGE	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	2	33.33	2	33.33
2	1	16.67	3	50.00
3	1	16.67	4	66.67
4	2	33.33	6	100.00

BGP can be used as an alternative to the CLASS statement in PROC MEANS to compute summary statistics for groups of data rather than for an entire data set. When the CLASS statement is used, the data set need not be sorted by the variable(s) in the statement. When a BY statement is used, just as with PROC PRINT and PROC FREQ, the data set must be sorted.

...Example 6.33...

```
proc sort data=x.cancer99 (keep=gender age) out=all; ①
by gender;
run;

title 'PROC MEANS WITH A BY VARIABLE';
proc means data=all maxdec=1;
var age;
by gender; ②
run;

title 'PROC MEANS WITH A CLASS VARIABLE';
proc means data=all maxdec=1;
var age;
class gender; ③
run;
```

PROC MEANS WITH A BY VARIABLE

GENDER=1

Analysis Variable : AGE AGE AT DEATH (YEARS)

N	Mean	Std Dev	Minimum	Maximum
18541	70.1	13.4	0.0	112.0

GENDER=2

Analysis Variable : AGE AGE AT DEATH (YEARS)

N	Mean	Std Dev	Minimum	Maximum
19037	71.0	14.2	0.0	108.0

PROC MEANS WITH A CLASS VARIABLE

Analysis Variable : AGE AGE AT DEATH (YEARS)

GENDER	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	18542	18541	70.1	13.4	0.0	112.0
2	19041	19037	71.0	14.2	0.0	108.0

All observations in data set CANCER99 are sorted by the variable GENDER, but only the variables GENDER and AGE are placed in the new data set ALL ①. PROC MEANS is used twice, the first time with the variable GENDER in a BY statement ②, then in a CLASS statement ③. The values of the statistics in the above output are the same for both uses of PROC MEANS. Only the appearance of the output is different.

Later chapters show how BGP is used to in other procedures and data steps. One of the most common uses of BGP is within a data step and merging two or more data sets. You should remember that before you use a variable in a BY statement in a procedure (or as you will see in later chapters, in a data step), the data set must be sorted according to the variable(s) in the BY statement.

NOTE: BY-group processing requires that data be sorted in ascending order. This is not always true in that there are options to override this rule.

...PROC REPORT/REPORTING...

PROC REPORT has many more reporting capabilities than PROC PRINT, so many that it merits its own manual, multiple books, and numerous papers given at SAS conferences. The SAS procedures manual describes PROC REPORT as follows...

The REPORT procedure combines features of the PRINT, MEANS, and TABULATE procedures with features of the DATA step in a single report-writing tool that can produce a variety of reports.

PROC REPORT is an interactive procedure in that can be run in both interactive and batch mode. Interactive mode opens a REPORT window in addition to the standard EDITOR/LOG/OUTPUT windows in the SAS display manager. In batch mode, you write and submit statements as with most other SAS procedures. All examples shown here specify the NOWD (no window) option and use PROC REPORT in batch mode.

The default behavior of PROC REPORT is to print all variables and all observations. If variables are formatted, the formatted values are printed. Only a few of the features of PROC REPORT are shown in the following examples.

DEFAULT OUTPUT OF PROC REPORT

GAZETTEER		AGE AT DEATH (YEARS)	ICD-10 CAUSE OF DEATH	WHERE DEATH
33	2	79	C56	B
33	2	77	C509	D
33	1	72	C349	D
33	2	79	C183	D
33	1	52	C349	F
33	2	73	C259	F
05	1	76	C819	D
33	1	65	C349	D
33	1	80	C64	F
26	2	45	C449	D

...Example 6.34...

```
title 'DEFAULT OUTPUT OF PROC REPORT';
proc report data=x.cancer99 (firtsobs=501 obs=510) nowd; ①
run;
```

PROC REPORT is used in batch mode (notice the NOWD option) to display ten observations ①. If you compare the output on the right to that produced bt PROC PRINT in example 6.5, you will notice several differences: variable labels are used by default as the column headers (in PROC PRINT, variable names are used); the width of columns for character variables is determined by variable length (in PROC PRINT, the length of the variable name or label is used in determining column width); the default width of columns for numeric variables is 9 (in PROC PRINT, the length of the variable name or label is used in determining column width); there are no observation numbers (in PROC PRINT, observation numbers are displayed unless suppressed using the NOOBS option). There is another major difference between the REPORT and PRINT procedures and it occurs when all you want to display is the value of numeric variables.

...Example 6.35...

```
title 'PROC REPORT, NUMERIC VARIABLES ONLY';
proc report data=x.cancer99 (keep=age) nowd; ①
run;
```

A KEEP data set option restricts PROC REPORT output to the only numeric variable in the data set, AGE ①. The output on the right shows that default in PROC REPORT for numeric data is to display the sum of the variable values across all observations. PROC PRINT would display the value of the variable AGE in all observations (many lines of output).

PROC REPORT, NUMERIC VARIABLES ONLY

AGE AT DEATH (YEARS)
2651976

Given the default output displayed above, you might wonder why you would use PROC REPORT. If all you want to do is take a quick look at your data, PROC PRINT is probably a better choice. The next few examples show you when PROC REPORT might be a better alternative.

Before showing some suggested uses of PROC REPORT, you should know a few other way that it differs frpm PROC PRINT.

...Example 6.36...

```
title 'USING COLUMN AND DEFINE STATEMENTS';
proc report data=x.cancer99 (firstobs=501 obs=510) nowd; ①
columns age;  ②
define age / display;  ③
run;
```

Observations for display are limited using data set options ①. PROC REPORT uses a COLUMNS statement (not a VAR statement as used in PROC PRINT) to list the variables to be displayed ②. AGE is a numeric variable and without any further statements, the output would show the sum of the variable AGE across observations 501 through 510. A DEFINE statement uses a DISPLAY option to tell SAS that you want values of the variable displayed rather than just calculating a sum ③. By default, PROC REPORT treats numeric variables as ANALYSIS variables, meaning that you want statistics calculated and the default statistic is the sum. Character variables are treated as DISPLAY variables by default. There is no statement in PROC PRINT that is the equivalent of the DEFINE statement in PROC REPORT.

USING COLUMN AND DEFINE STATEMENTS

AGE AT DEATH (YEARS)
79
77
72
79
52
73
76
65
80
45

...Example 6.37...

```
title 'MORE USES OF DEFINE STATEMENTS';
proc report data=x.cancer99 (firstobs=501 obs=510) nowd;
columns county age cause; ①
define county / center width=9 'RESIDENCE COUNTY'; ②
define cause  / center width=6; ③
define age    / center width=7 format=3.; ④
run;
```

The COLUMNS statement specifies both variables to be used and their order in PROC REPORT output ①. DEFINE statements are used for all three variables. For COUNTY, the values (and column header) are centered, the column width is changed from 2 (value width) to 9, and the column header is changed (note, the width of 9 is the length of the word 'RESIDENCE') ②. The display of the CAUSE is centered and the column with is changed from 4 (value width) to 6 (width of 'ICD-10' in the variable label) ③. Values of the numeric variable AGE are also centered, and the column width is changed from 9 (default for numeric data) to 7 ④. Notice that there is also a FORMAT option in the last DEFINE statement. Without that option, the values of AGE would be left-justified within the width of the column display, though the column header would be centered. To center the values of numeric variables, you need the FORMAT option to limit the number of columns used for value display. You should also note that the DEFINE statements need not be in the same order as the variables in COLUMNS statement.

MORE USES OF DEFINE STATEMENTS

RESIDENCE COUNTY	AGE AT DEATH (YEARS)	ICD-10 CAUSE OF DEATH
33	79	C56
33	77	C509
33	72	C349
33	79	C183
33	52	C349
33	73	C259
05	76	C819
33	65	C349
33	80	C64
26	45	C449

Now that you know some of the default behaviors of PROC REPORT and a little about using COLUMNS and DEFINE statements to override the defaults, there are some nice features of PROC REPORT that make it a good alternative to PROC PRINT. The first of these is the PANELS option.

...Example 6.38...

```
title 'THE PANELS OPTION (MAXIMUM PAPER USE)';
proc report data=x.cancer99 panels=99 nowd;  ①
where county eq '01';  ②
columns age gender cause;
define age    / center width=7 format=3.;
define gender / center width=6;
define cause  / center width=6;
run;
```

THE PANELS OPTION (MAXIMUM PAPER USE)

AGE AT DEATH (YEARS)	GENDER	ICD-10 CAUSE OF DEATH	AGE AT DEATH (YEARS)	GENDER	ICD-10 CAUSE OF DEATH	AGE AT DEATH (YEARS)	GENDER	ICD-10 CAUSE OF DEATH
70	1	C349	89	2	C959	70	1	C61
62	2	C189	88	2	C80	62	1	C349
36	2	C56	86	1	C64	50	2	C539
61	1	C911	69	1	C439	82	1	C61
45	2	C80	76	1	C349	85	1	C19
88	2	C259	76	2	C20	55	2	C349
76	2	C539	85	1	C20	53	2	C240
82	2	C349	78	1	C61	55	2	C349
50	1	C833	54	2	C259	78	1	C449
69	2	C80	93	2	C80	93	2	C349
63	1	C349	65	2	C349	71	1	C349
68	1	C169	68	1	C719	92	2	C64
80	2	C259	64	1	C349	79	2	C349
73	2	C80	90	2	C509	65	1	C169
73	1	C220	76	1	C80	77	1	C349
69	1	C19	76	2	C859	76	2	C80
68	1	C189	79	2	C169	68	1	C189
73	1	C859	90	1	C61	79	1	C459

The PANELS option is added to PROC REPORT ① and output is restricted to only those observations from county '01' ② . There are three variables in the columns statement. Without the PANELS option, there would be three columns of data on each page of output, as there would be in PROC PRINT. The PANELS option allows you to ask for multiple sets the columns to be displayed on each page of output. The PANELS option can be used any time that the regular output would take up less than half the page width. If you want the procedure to determine the number of panels that maximize the use of space on a page, use a large number. Though the option says PANELS=99, the above output shows that three panels fit in the width of a page. Only a portion of an entire page of output is displayed.

Another useful feature of PROC REPORT is the ability to display observations in sorted order without having to sort the data set.

...Example 6.39...

```
title 'THE PANELS OPTION PLUS SPECIFIED ORDER';
proc report data=x.cancer99 panels=99 nowd;
where county eq '01';
columns age gender cause;
define age    / center width=7 format=3. order; ①
define gender / center width=6 order; ②
define cause  / center width=6;
run;
```

THE PANELS OPTION PLUS SPECIFIED ORDER

AGE AT DEATH (YEARS)	GENDER	ICD-10 CAUSE OF DEATH	AGE AT DEATH (YEARS)	GENDER	ICD-10 CAUSE OF DEATH	AGE AT DEATH (YEARS)	GENDER	ICD-10 CAUSE OF DEATH
4	2	C419		2	C349			C64
36	2	C56	52	1	C349			C509
39	1	C229		2	C419	58	1	C349
		C859			C179			C760
	2	C509			C64			C349
40	2	C910	53	1	C80			C349
41	1	C64			C900			C029
	2	C509		2	C240			C349
42	1	C921			C189	59	1	C911
43	1	C349			C969			C189
		C220	54	1	C259			C349
	2	C509			C349		2	C80
44	1	C509			C349			C349
		C80			C140			C920
	2	C349		2	C12			C349
45	1	C349			C259	60	1	C900
		C349			C349			C760
		C349			C170			C64
		C20			C509			C349

The variable AGE is defined as an ORDER variable ①. Remember that numeric variables are considered as ANALYSIS variables and character variables are DISPLAY variables. In a example 6.36, a DEFINE statement was used to make the NUMERIC variable AGE a DISPLAY variable. The character variable GENDER is also defined as an ORDER variable ②. In the above out, the data are displayed in ascending sorted order by AGE and by GENDER within AGE. However, PROC SORT was not used to sort the data prior to PROC REPORT as you would have to do with PROC PRINT.

The last feature to be illustrated is the being able to display both the value of a variable and its formatted value. PROC FORMAT is discussed in chapter 7, but user-defined formats have been introduced in a few examples in this chapter.

...Example 6.40...

```
proc format;
value $cause ①
'C15' - 'C159' = 'ESOPHAGUS'
'C18' - 'C189' = 'COLON'
'C25' - 'C259' = 'PANCREAS'
'C34' - 'C349' = 'LUNG'
'C50' - 'C509' = 'BREAST'
'C56'          = 'OVARY';
run;

title 'USING A VARIABLE ALIAS';
proc report data=x.cancer99 panels=99 pspace=10 nowd; ②
where county eq '01' and cause in : ('C15' 'C18' 'C25' 'C34' 'C50' 'C56'); ③
columns age gender cause cause=c_name; ④
define age    / center width=7 format=3. order;
define gender / center width=6 order;
define cause  / center width=6;
define c_name / ' ' format=$cause. ; ⑤
run;
```

USING A VARIABLE ALIAS

AGE AT DEATH (YEARS)	GENDER	ICD-10 CAUSE OF DEATH		AGE AT DEATH (YEARS)	GENDER	ICD-10 CAUSE OF DEATH	
36	2	C56	OVARY			C341	LUNG
39	2	C509	BREAST			C259	PANCREAS
41	2	C509	BREAST		2	C349	LUNG
43	1	C349	LUNG			C509	BREAST
	2	C509	BREAST			C509	BREAST
44	1	C509	BREAST	58	1	C349	LUNG
	2	C349	LUNG			C349	LUNG
45	1	C349	LUNG			C349	LUNG
		C349	LUNG			C349	LUNG
		C349	LUNG	59	1	C189	COLON
		C159	ESOPHAGUS			C349	LUNG
	2	C349	LUNG		2	C349	LUNG
		C56	OVARY			C349	LUNG
46	2	C509	BREAST	60	1	C349	LUNG
47	1	C259	PANCREAS			C349	LUNG
48	1	C259	PANCREAS			C180	COLON
		C349	LUNG		2	C509	BREAST
	2	C509	BREAST			C56	OVARY
49	1	C349	LUNG	61	1	C349	LUNG

PROC FORMAT creates a set of rules that can be used label the values of the variable cause with a literal description ①. The PANELS option is used again, but this time the PSPACE option is also used to define the number of spaces that should occur between panels ②. In addition to limiting observations to those from county '01', observations are also restricted to six causes (the six causes listed in PROC FORMAT) ③. The COLUMNS statement uses the variable alias feature (CAUSE=C_NAME) of PROC REPORT ④. Just for the duration of PROC REPORT, a new variable is available, C_NAME, that has the same value as the variable CAUSE. A DEFINE statement asks that the new variable be displayed using the format that converts a cause of death to a literal. There is no variable alias feature in PROC PRINT. You can display either the value of a variable or its formatted value, but not both.

The last PROC REPORT example shows yet another feature that is not available in PROC PRINT, the BOX option.

...Example 6.41...

```
proc format;
value $cause
'C15' - 'C159' = 'ESOPHAGUS'      'C18' - 'C189' = 'COLON'
'C25' - 'C259' = 'PANCREAS'       'C34' - 'C349' = 'LUNG'
'C50' - 'C509' = 'BREAST'         'C56'          = 'OVARY';
value $place ①
'A'-'C'     = 'HOSPITAL (OTHER)'  'D' = 'HOSPITAL INPT'
'E'         = 'OTHER INST'              'F' = "DECEDENT'S RES"
'G','H','N' = 'OTHER NON-HOSP'          'Z' = 'UNKNOWN';
run;

title 'USING VARIABLE ALIASES AND THE BOX OPTION';
proc report data=x.cancer99 box nowd; ②
where county eq '01' and cause in : ('C15' 'C18' 'C25' 'C34' 'C50' 'C56');
columns age gender cause cause=c_name place place=p_name; ③
define age    / center width=7 format=3. order;
define gender / center width=6 order;
define cause  / center width=6;
define c_name / ' ' format=$cause.;  ④
define place  / center width=5 'PLACE';
define p_name / ' ' format=$place.;  ⑤
run;
```

Another user-defined format is added in PROC FORMAT, $PLACE ①. This set of rules can be used with the variable PLACE to display a literal describing where death occurred. The PANELS option is removed and the BOX option is used ②. As you can see on the right, this option adds boxes around the displayed values of variables. A new variable alias (PLACE=P_NAME) is added to the columns statement ③. The two variable aliases are used to add literal descriptions of the variable values to the displayed output.

USING VARIABLE ALIASES AND THE BOX OPTION

AGE AT DEATH (YEARS)	GENDER	ICD-10 CAUSE OF DEATH		PLACE	
36	2	C56	OVARY	F	DECEDENT'S RES
39	2	C509	BREAST	D	HOSPITAL INPT
41	2	C509	BREAST	F	DECEDENT'S RES
43	1	C349	LUNG	E	OTHER INST
	2	C509	BREAST	D	HOSPITAL INPT
44	1	C509	BREAST	F	DECEDENT'S RES
	2	C349	LUNG	E	OTHER INST
45	1	C349	LUNG	D	HOSPITAL INPT
		C349	LUNG	E	OTHER INST
		C349	LUNG	E	OTHER INST
		C159	ESOPHAGUS	D	HOSPITAL INPT
	2	C349	LUNG	F	DECEDENT'S RES
		C56	OVARY	C	HOSPITAL (OTHER)

Hopefully you now understand the worth of PROC REPORT. The features described in the examples are just a few of those that you can use, but they are very useful. Use PROC PRINT for simple looks at your data, use PROC REPORT when you need some features that you cannot find in PROC PRINT.

...PROC UNIVARIATE/DESCRIPTIVE STATISTICS

Though PROC MEANS can produce a number of descriptive statistics, PROC UNIVARIATE is also easy to use and produces a more comprehensive description of the numeric variables in a data set. A portion of the description of the procedure in SAS online help is as follows ...

The UNIVARIATE procedure provides the following: descriptive statistics based on moments (including skewness and kurtosis), quantiles or percentiles (such as the median), frequency tables, and extreme values; histograms and comparative histograms; goodness-of-fit tests for a variety of distributions including the normal; the ability to inset summary statistics on plots produced on a graphics device; the ability to analyze data sets with a frequency variable; the ability to create output data sets containing summary statistics, histogram intervals, and parameters of fitted curves.

The default behavior of PROC UNIVARIATE is to produce descriptive statistics for all numeric variables using all observations with non-missing values. The default output includes moments, basic statistical measures, tests for location, quantiles, and information about both extreme and missing values.

...Example 6.42...

```
title 'DEFAULT OUTPUT OF PROC UNIVARIATE';
proc univariate data=x.cancer99; ①
run;

title 'ADD PLOTS AND TEST Ho: AGE = 70';
proc univariate data=x.cancer99 mu0=70 plots normal; ②
var age; ③
ods select testsforlocation testsfornormalityplots; ④
run;
```

PROC UNIVARIATE is run without any options ①. The procedure will compute descriptive statistics on the only numeric variable in the data set, AGE. In the output on the next page, you can see a section labeled 'TESTS FOR LOCATION'. These are the results of testing the mean to see if it is equal to zero. That test can be changed using the MU0 option to a test using a different null value, in this example age 70 ②. Also, several plots can be added to the output (option PLOTS) as can several tests for normality

(option NORMAL). Though not needed here, you can select analysis variables with a VAR statement ③. The output delivery system (ODS) allows you to select portions of the output from any SAS procedure. An ODS SELECT statement specifies that we want only three parts of univariate output ④.

The default output from PROC UNIVARIATE is quite comprehensive. As you can see on the right, in addition to a number of descriptive statistics, the five highest and lowest values in the data set are listed with a count of the number of missing values. The *Tests for Location* in the middle of the output present both parametric (t-test) and non-parametric (Sign and Signed-Rank) tests of the mean versus a null value of zero. The output below shows the results of: restricting the univariate output to only three sections; changing the null value of the tests for location from 0 to 70; adding normality test and plots.

DEFAULT OUTPUT OF PROC UNIVARIATE

Variable: AGE (AGE AT DEATH (YEARS))

Moments

N	37578	Sum Weights	37578
Mean	70.5725691	Sum Observations	2651976
Std Deviation	13.8227372	Variance	191.068063
Skewness	-0.8225036	Kurtosis	1.22025703
Uncorrected SS	194336524	Corrected SS	7179764.6
Coeff Variation	19.586558	Std Error Mean	0.07130619

Basic Statistical Measures

Location		Variability	
Mean	70.57257	Std Deviation	13.82274
Median	73.00000	Variance	191.06806
Mode	73.00000	Range	112.00000
		Interquartile Range	18.00000

Tests for Location: Mu0=0

Test	-Statistic-		-----p Value-----	
Student's t	t	989.7117	Pr > |t|	<.0001
Sign	M	18785.5	Pr >= |M|	<.0001
Signed Rank	S	3.529E8	Pr >= |S|	<.0001

Quantiles (Definition 5)

Quantile	Estimate
100% Max	112
99%	95
95%	90
90%	86
75% Q3	80
50% Median	73
25% Q1	62
10%	52
5%	46
1%	32
0% Min	0

Extreme Observations

----Lowest----		----Highest---	
Value	Obs	Value	Obs
0	12	103	37579
0	11	103	37580
0	10	105	37581
0	9	108	37582
0	8	112	37583

Missing Values

Missing Value	Count	-----Percent Of----- All Obs	Missing Obs
.	5	0.01	100.00

ADD PLOTS AND NORMALITY TEST, CHANGE LOCATION TEST TO Ho: AGE = 70

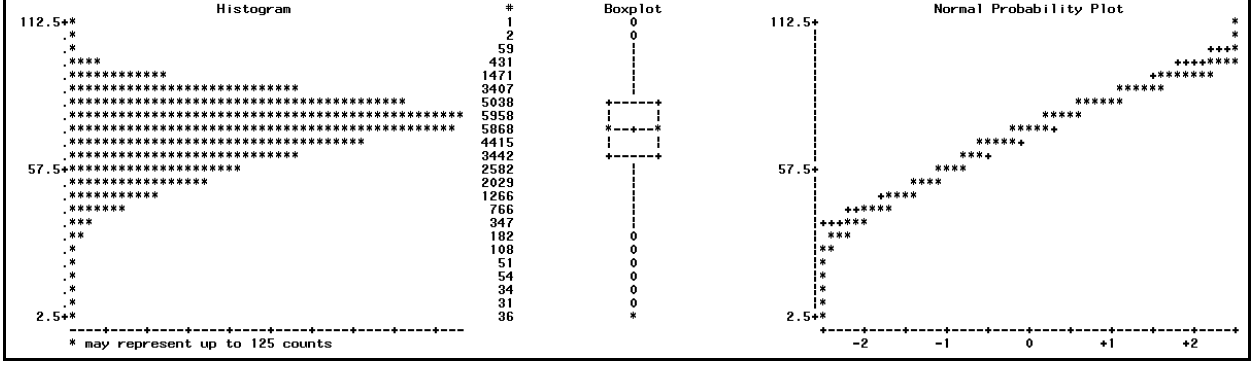
Variable: AGE (AGE AT DEATH (YEARS))

Tests for Location: Mu0=70

Test	-Statistic-		-----p Value-----	
Student's t	t	8.029725	Pr > |t|	<.0001
Sign	M	2925.5	Pr >= |M|	<.0001
Signed Rank	S	39245306	Pr >= |S|	<.0001

Tests for Normality

Test	--Statistic---		-----p Value------	
Kolmogorov-Smirnov	D	0.076334	Pr > D	<0.0100
Cramer-von Mises	W-Sq	48.58488	Pr > W-Sq	<0.0050
Anderson-Darling	A-Sq	283.0582	Pr > A-Sq	<0.0050



One of the uses of PROC UNIVARIATE is the identification of possible outliers, also referred to as extreme observations. By default, PROC UNIVARIATE lists the five highest and lowest values of numeric variables. That default can be changed and you can request counts of the number of extreme values rather than just a listing.

...Example 6.43...

```
title 'IDENTIFY EXTREME OBSERVATIONS';
proc univariate data=x.cancer99 nextrobs=10; ①
id gender cause; ②
ods select extremeobs; ③
run;

title 'COUNT EXTREME OBSERVATIONS';
proc univariate data=x.cancer99 nextrval=10; ④
ods select extremevalues; ⑤
run;
```

The number of extreme observations to be listed is raised from the default of 5 to 10 ①. An ID statement specifies variables whose values will also appear in the extreme value listing ②. Normally, only the extreme values and observations numbers are listed. Now, you can see both the gender and cause of death associated with the extreme values of the variable AGE. An ODS SELECT statement limits the output to only the extreme values ③.

IDENTIFY EXTREME OBSERVATIONS

Variable: AGE (AGE AT DEATH (YEARS))

Extreme Observations

Lowest				Highest			
Value	GENDER	CAUSE	Obs	Value	GENDER	CAUSE	Obs
0	1	C80	12	102	1	C61	37574
0	1	C439	11	103	2	C169	37575
0	1	C798	10	103	2	C189	37576
0	2	C719	9	103	1	C760	37577
0	2	C80	8	103	1	C189	37578
0	2	C719	7	103	1	C189	37579
0	2	C959	6	103	1	C169	37580
1	1	C80	18	105	2	C189	37581
1	1	C859	17	108	2	C189	37582
1	2	C80	16	112	1	C349	37583

COUNT EXTREME OBSERVATIONS

Variable: AGE (AGE AT DEATH (YEARS))

Extreme Values

Lowest			Highest		
Order	Value	Freq	Order	Value	Freq
1	0	7	98	97	82
2	1	6	99	98	55
3	2	8	100	99	30
4	3	6	101	100	23
5	4	9	102	101	19
6	5	8	103	102	11
7	6	6	104	103	6
8	7	3	105	105	1
9	8	5	106	108	1
10	9	9	107	112	1

You can also produce counts of extreme values if you use the NEXTRVAL (number of extreme values) option ④. PROC UNIVARIATE is instructed to list the frequency of the ten highest and lowest values of AGE in the data set. Output is once again limited to only the extreme values, but this time there are counts rather than a listing ⑤. In the listing of extreme values in the upper table, you can see all the ages with a value of 0, but do not know that there are six observations in the data set with an with an age of 1 -- easily seen in the lower table. Both types of output are useful and you can combine the requests for such output in one run of PROC UNIVARIATE rather than make two passes through the data as was done in example 6.43.

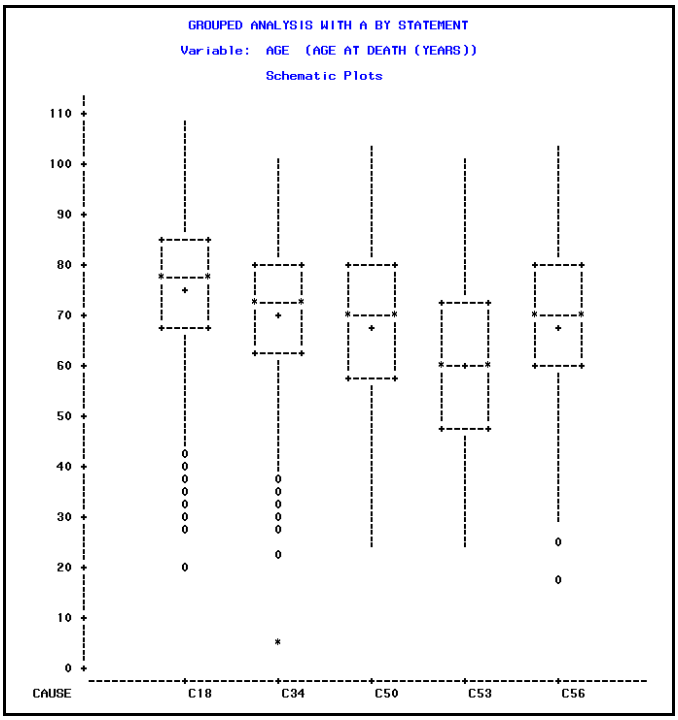
Just as a CLASS statement was used in PROC MEANS to produce an analysis with groups, it can also be used in PROC UNIVARIATE. The only difference is that in PROC UNIVARIATE you are limited to two variables in the CLASS statement, while in PROC MEANS there is no limit. A BY statement can be used in place of the CLASS statement to perform analyses with groups, but the data set must first be sorted according to values of the by-variable(s).

...Example 6.44...

```
title 'GROUPED ANALYSIS WITH A CLASS STATEMENT';
proc univariate data=x.cancer99;
class gender; ①
ods select basicmeasures; ②
run;
```

```
proc sort data=x.cancer99 out=ftemp; ③
by cause; ④
where cause in: ('C18' 'C34' 'C50' 'C53' 'C56') and gender eq '2'; ⑤
run;

title 'GROUPED ANALYSIS WITH A BY STATEMENT';
proc univariate data=ftemp plot; ⑥
by cause; ⑦
format cause $3.; ⑧
ods select ssplots; ⑨
run;
```



GROUPED ANALYSIS WITH A CLASS STATEMENT

Variable: AGE (AGE AT DEATH (YEARS))
GENDER = 1

Basic Statistical Measures

Location		Variability	
Mean	70.08505	Std Deviation	13.39028
Median	72.00000	Variance	179.29951
Mode	73.00000	Range	112.00000
		Interquartile Range	17.00000

GROUPED ANALYSIS WITH A CLASS STATEMENT

Variable: AGE (AGE AT DEATH (YEARS))
GENDER = 2

Basic Statistical Measures

Location		Variability	
Mean	71.04738	Std Deviation	14.21559
Median	73.00000	Variance	202.08307
Mode	73.00000	Range	108.00000
		Interquartile Range	18.00000

A CLASS statement request that analysis be done with observations grouped according to values of the variable GENDER ①. The output is limited to basic s