

Chapter 1

Introduction to SAS

Arthur Li

SAS Program and Language

```
/*  
An example  
*/  
data hearing;  
    infile "U:\Individual\hearing_simple.txt";  
    input id    $ 1 - 4  fname  $ 6 - 20  
          lname  $ 23 - 33 race   $ 35  
          smoke  $ 37 - 43 Age  45 - 46  
          Preg  48 Hearing $ 50 - 52  
          income    54-59;  
  
run;  
proc print data=hearing;  
run;  
  
*The Contents Procedure;  
proc contents data=work._ALL_;  
run;  
  
*The FREQ Procedure;  
proc freq data=hearing;  
    tables preg;  
run;
```

SAS Program and Language

```
/*  
An example  
*/  
data hearing;  
    infile "U:\Individual\hearing_simple.txt";  
    input id    $ 1 - 4   fname  $ 6 - 20  
          lname  $ 23 - 33 race   $ 35  
          smoke  $ 37 - 43 Age    45 - 46  
          Preg   48 Hearing $ 50 - 52  
          income    54-59;  
  
run;  
proc print data=hearing;  
run;  
  
*The Contents Procedure;  
proc contents data=work._ALL_;  
run;  
  
*The FREQ Procedure;  
proc freq data=hearing;  
    tables preg;  
run;
```

- ❖ SAS program is made-up from SAS statements
- ❖ Each SAS statement ends with a (;)
- ❖ Case Insensitive
- ❖ Some statements can be used in both DATA and PROC steps

SAS Program and Language

❖ Other types of SAS language elements:

- ☐ Data set options
- ☐ Expressions
- ☐ Formats
- ☐ Informats

❖ These language elements are mostly used within a statement

Reading Data into SAS

- ❖ The starting point: reading data into the SAS system.
- ❖ A SAS data set is often created by reading, extracting, or combining data from
 - ❑ SAS data sets
 - ❑ raw text file
 - ❑ EXCEL file
- ❖ A created SAS data set is often saved as a permanent file:
 - ❑ Reading a SAS data set is simple
 - ❑ A SAS data set is easy to manage

The SAS Data Set and SAS Library

Observations

	ID	AGE	GROUP
1	M2390	30	P
2	F2390	.	D
3	F2340	26	
4	M1240	12	D

Variables

- ❖ Character variables: alphabetic characters, 0 - 9, special characters
- ❖ Numeric variables: floating-point numbers, including dates and times

The SAS Data Set and SAS Library

- ❖ A SAS data set also contains descriptor information
 - ❑ variable attributes: name, length, type, label, format, informat, etc
 - ❑ when the data is being created
 - ❑ number of observations, etc

The SAS Data Set and SAS Library

- ❖ A SAS data is stored in a SAS library
- ❖ A SAS library is a folder in which SAS files are stored
- ❖ To access/create a SAS file in the library, use the LIBNAME statement

The SAS Data Set and SAS Library

❖ The LIBNAME statement:

```
LIBNAME libref 'SAS-library';
```

❖ A LIBREF is a name that you associate with the physical location of the SAS library

❖ Rules for naming the LIBREF

- ❑ ≤ 8 characters

- ❑ must begin with either an _ or letters

- ❑ can only contain letters, numbers, and _

The SAS Data Set and SAS Library

❖ The LIBNAME statement:

```
LIBNAME libref 'SAS-library';
```

- ❖ 'SAS-LIBRARY' is the physical location name
- ❖ Once the LIBNAME statement is submitted, instead of referring to the file's directory with the complete path name, you will use LIBREF.

The SAS Data Set and SAS Library

❖ The LIBNAME statement:

```
LIBNAME libref 'SAS-library';
```

```
libname saslib 'C:\SAS Book\dat';
```

The SAS Data Set and SAS Library

- ❖ The LIBNAME statement:

```
LIBNAME libref 'SAS-library';
```

- ❖ The LIBNAME statement is global; means the name of the library (LIBREF) is only in effect until you
 - ☐ change it,
 - ☐ cancel it, or
 - ☐ end your current SAS session
- ❖ The contents of the library always exists unless you delete it

The SAS Data Set and SAS Library

❖ A SAS data set has two level names:

`libref.filename`

❖ The rules for naming a SAS data set:

- ❑ ≤ 32 characters

- ❑ must begin with either an _ or letters

- ❑ can only contain letters, numbers, and _

The SAS Data Set and SAS Library

- ❖ A SAS data set has two level names:

`libref.filename`

- ❖ A SAS data set is either stored in a permanent or a temporary library
- ❖ WORK: a temporary library used to store your temporary files
- ❖ A temporary file only exists during the current SAS session

The SAS Data Set and SAS Library

- ❖ A SAS data set has two level names:

`libref.filename`

- ❖ To reference a permanent data set,
`saslib.noise`
- ❖ To reference a temporary data set,
`work.noise`
`noise`

Reading a SAS Data Set

- ❖ To read a SAS data set into SAS, you need to use at least the following three statements:

```
DATA output-data-set-name;  
    SET input-data-set-name;  
RUN;
```


Reading a SAS Data Set

Program 1.1

```
/*  
Program 1.1  
*/  
libname saslib 'W:\SAS Book\dat';  
libname desktop 'C:\Documents and Settings\Desktop';  
data noise1;    *creates NOISE1.SAS7BDAT in the library WORK;  
    set saslib.noise;  
run;  
  
data desktop.noise1; /*becomes NOISE1.SAS7BDAT on C:\Documents and  
Settings\Desktop */  
    set saslib.noise;  
run;
```

Reading a SAS Data Set

Log from Program 1.1

```
25  data noise1;    *creates NOISE1.SAS7BDAT in the library WORK;
26      set saslib.noise;
27  run;
```

NOTE: There were 32 observations read from the data set SASLIB.NOISE.

NOTE: The data set WORK.NOISE1 has 32 observations and 4 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.01 seconds

```
28
29  data desktop.noise1; /*becomes NOISE1.SAS7BDAT on w:\SAS Book\dat */
30      set saslib.noise;
31  run;
```

NOTE: There were 32 observations read from the data set SASLIB.NOISE.

NOTE: The data set DESKTOP.NOISE1 has 32 observations and 4 variables.

NOTE: DATA statement used (Total process time):

real time	0.00 seconds
cpu time	0.00 seconds

Reading a Raw Data File with Fixed Fields

hearing.txt

Raw Data fixed field

---5---0---5---0---5---0

629F	H	past	26	0		35000
656F	W	never	26	1	no	48000
711F	W	never	32	1	no	30000
511F	B	never	32	0		25000
478F	W	past	34	0		35700

Variable Name	Columns	Type
ID	1-4	Character
RACE	6	Character
SMOKE	8-14	Character
AGE	16,17	Numeric
PREG	19	Numeric
HEARING	21-23	Character
INCOME	25-30	Numeric

Reading a Raw Data File with Fixed Fields

❖ INFILE statement: specifying the location of the file:

INFILE file-specification <OBS=record-number>;

```
infile "W:\SAS Book\dat\hearing.txt";
```

Reading a Raw Data File with Fixed Fields

- ❖ INPUT statement
 - ☐ Column input
 - ☐ Formatted input
 - ☐ List input
 - ☐ Named input

Reading a Raw Data File with Fixed Fields

❖ Column INPUT method:

- ☐ data are in a fixed field
- ☐ contains characters/standard numeric values

❖ Standard numeric values:

- ☐ numbers
- ☐ decimal points
- ☐ numbers in scientific notation
- ☐ plus/minus signs

❖ Non-standard numeric values:

- ☐ date and time values
- ☐ fractions
- ☐ integer and real binary numbers in hexadecimal forms
- ☐ values contain: %, \$, and comma (,)

Reading a Raw Data File with Fixed Fields

❖ The INPUT statement (column input):

```
INPUT variable <$> start-column <- end-column>;
```

❑ The rules for naming the SAS variable is the same as naming the SAS data set

```
input id $ 1 - 4 race $ 6 smoke $ 8 - 14  
      Age 16 - 17 Preg 19 Hearing $ 21 - 23  
      Income 25 - 30;
```

Reading a Raw Data File with Fixed Fields

Variable Name	Columns	Type
ID	1-4	Character
RACE	6	Character
SMOKE	8-14	Character
AGE	16,17	Numeric
PREG	19	Numeric
HEARING	21-23	Character
INCOME	25-30	Numeric

Program 1.2

```
data hearing;  
    infile "W:\SAS Book\dat\hearing.txt";  
    input id $ 1 - 4   race $ 6   smoke $ 8 - 14  
          Age 16 - 17   Preg 19   Hearing $ 21 - 23  
          Income 25 - 30;  
run;
```


Reading Data Entered Directly into the Program

Program 1.3

```
data hearing_small;  
    input id $ 1 - 4 race $ 6 smoke $ 8 - 14  
          Age 16 - 17 Preg 19 Hearing $ 21 - 23  
          Income 25 - 30;  
datalines;  
629F H past      26 0      35000  
656F W never     26 1 no    48000  
711F W never     32 1 no    30000  
733F W current   17 0      59000  
135F B current   29 1 no    120000  
982F W past      26 1 yes   113000  
;
```

❖ If the data contains a semicolon

❑ use DATALINES4 instead of DATALINES

❑ use 4 consecutive semicolons (;;;;) instead of 1 at the end

Creating and Modifying Variables

Assignment Statement and SAS Functions

- ❖ Creating and modifying variables is one of the most important programming tasks

- ❖ Assignment statement:

```
variable = expression;
```

- ❖ VARIABLE is either a new or existing variable, and

- ❖ EXPRESSION is any valid SAS expression.

Assignment Statement and SAS Functions

- ❖ Using an expression in a SAS statement is to ...
 - ❑ create variables
 - ❑ assign values,
 - ❑ perform calculations,
 - ❑ transform variables
 - ❑ perform conditional processing.

- ❖ All expressions will return a result with a character, numeric, or Boolean value

Assignment Statement and SAS Functions

- ❖ An expression is formed with operands + operators.
- ❖ Operands: constants or variables.
- ❖ Operators: symbols for arithmetic calculations, comparisons, logical operations, SAS functions, or grouping parentheses.
- ❖ Examples of operators, along with their evaluation orders, are illustrated in Table 1.2.

Assignment Statement and SAS Functions

- ❖ Types of operators:
 - ❑ Arithmetic (+, -, **, etc.)
 - ❑ Comparison (>, >=, etc.)
 - ❑ Logical (&, |, +)

- ❖ Comparison operators are often used with IF-THEN/ELSE statements.

- ❖ The result from a comparison is 1 or 0

Assignment Statement and SAS Functions

- ❖ An expression can be categorized into simple, compound, and WHERE expression
- ❖ A simple expression: no more than 1 operator
- ❖ A compound expression: > 1 operators
- ❖ Connect one or more simple expressions with logical operators → compound expression:

a<10 & a>5

Assignment Statement and SAS Functions

- ❖ A SAS function can also be considered an operator:

`function-name (argument-1<,...argument-n>)`

- ❖ The SUM function:

SUM (argument-1<,...argument-n>)

Assignment Statement and SAS Functions

Program 1.4

```
data score;
    input ID $ 1-4 score1 6-7 score2 9-10 score3 12-13;
    score_sum1 = score1 + score2 + score3;
    score_sum2 = sum(score1, score2, score3);
datalines;
629F  5  6  9
656F  6 10  9
711F  0  .  3
511F  9  4 10
478F  .  5  3
;

title 'Adding Three Scores By Using + Operator and SUM Function';
proc print data=score;
run;
```


Assignment Statement and SAS Functions

Adding Three Scores By Using + Operator and SUM Function

Obs	ID	score1	score2	score3	score_ sum1	score_ sum2
1	629F	5	6	9	20	20
2	656F	6	10	9	25	25
3	711F	0	.	3	.	3
4	511F	9	4	10	23	23
5	478F	.	5	3	.	8

Creating Variables Conditionally

- ❖ Create variables conditionally by using the IF-THEN/ELSE statement:

```
IF expression THEN statement;  
<ELSE statement;>
```

- ❖ The EXPRESSION can be any valid SAS expression
- ❖ If the EXPRESSION is evaluated to be true, the IF-THEN statement executes the statement after the keyword THEN.
- ❖ If there is an optional ELSE statement and if the expression is evaluated to be false, the ELSE statement the ELSE statement is executed.

Creating Variables Conditionally

❖ Create a variable OVER10K:

- ❑ If income > 10,000, OVER10K = 1;
- ❑ otherwise OVER10K = 0

```
if income > 10000 then over10k = 1;  
if income <= 10000 then over10k = 0;
```

❖ The 2nd statement is executed even if the condition in the 1st statement is true.

Creating Variables Conditionally

❖ A more efficient way:

```
if income > 10000 then over10k = 1;  
else over10k = 0;
```

❖ The danger of above statement:

- ❑ A missing value is the smallest value.
- ❑ The observations with the missing values will be assigned to 0

Base SAS Procedures

❖ Three types of Base SAS Procedures:

- ❑ Report writing: display information, such as data, summary, or graphical reports, PRINT, MEANS, FREQ, etc
- ❑ Statistics: perform basic statistical computations, CORR, UNIVARIATE, MEANS, STANDARD, etc
- ❑ Utilities: perform utility operations, SORT, FORMAT, IMPORT, COMPARE, CONTENTS, etc

Common Statements in SAS Procedures

- ❖ The default title for a SAS procedure:
The SAS System

```
TITLE <'text' | "text">;
```

- ❖ Once a TITLE statement is submitted, it is used for all subsequent output.
- ❖ To cancel any previous title:

```
TITLE;
```

Common Statements in SAS Procedures

- ❖ Using the BY statement will categorize the output by each level of the variable(s):

```
BY <DESCENDING> variable-1  
    <...<DESCENDING> variable-n>;
```

- ❖ The VARIABLE in the BY statement: BY variable.
- ❖ Procedures supported the BY statement: REPORT, SORT (required), COMPARE, CORR, FREQ, MEANS, TRANSPOSE, PRINT, etc.

Common Statements in SAS Procedures

- ❖ The WHERE statement is used to subset the input data set by specifying some conditions.

WHERE where-expression;

- ❖ Procedures that supported the WHERE statement: REPORT, COMPARE, SORT, CORR, FREQ, MEANS, TRANSPOSE, PRINT, UNIVARIATE, etc

```
where gender = 'M' ;
```

```
where age>50 and smoke='never' ;
```

```
where race='W' or race='B' ;
```


The CONTENTS Procedure

- ❖ To see the descriptor portion of a SAS data set or the contents of the directory of a SAS library:

```
PROC CONTENTS <DATA=SAS-file-specification>  
                <VARNUM>;  
RUN;
```

- ❖ FILE-SPECIFICATION in the DATA= option can be:

```
<libref.>SAS-data-set
```

```
<libref.>_ALL_
```

- ❖ The default library: WORK
- ❖ VARNUM: prints the variable names by their created order

The CONTENTS Procedure

Program 1.5

```
title 'The Contents of Hearing Data';  
proc contents data=hearing varnum;  
run;
```

The Contents of Hearing Data

The CONTENTS Procedure

Data Set Name	WORK.HEARING	Observations	34
Member Type	DATA	Variables	7
Engine	V9	Indexes	0
Created	Monday, May 20, 2013 06:50:54 PM	Observation Length	40
Last Modified	Monday, May 20, 2013 06:50:54 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

The CONTENTS Procedure

Engine/Host Dependent Information

Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	34
Number of Data Set Repairs	0
Filename	C:\Users\Arthur\AppData\Local\Temp\ SAS Temporary Files_TD3508_ARTHURHOME- PC_\hearing.sas7bdat
Release Created	9.0301M0
Host Created	X64_7HOME

Variables in Creation Order

#	Variable	Type	Len
1	id	Char	4
2	race	Char	1
3	smoke	Char	7
4	Age	Num	8
5	Preg	Num	8
6	Hearing	Char	3
7	Income	Num	8

The SORT Procedure

```
PROC SORT <DATA=SAS-data-set>  
          <OUT=SAS-data-set>;  
  BY <DESCENDING> variable-1  
    <...<DESCENDING> variable-n>;  
RUN;
```

Program 1.6

```
proc sort data=hearing out=hearing_sort;  
  by race descending preg descending age ;  
run;
```

The PRINT Procedure

```
PROC PRINT <DATA=SAS-data-set> <NOOBS> ;  
  BY <DESCENDING> variable-1  
    <...<DESCENDING> variable-n>;  
  VAR variable(s);  
  WHERE where-expression;  
RUN;
```

Program 1.7

```
proc sort data=hearing_small out=hearing_small_sort;  
  by race;  
run;  
  
title 'Print ID SMOKE and AGE variables by RACE';  
proc print data=hearing_small_sort noobs;  
  by race;  
  var id smoke age;  
run;
```

The PRINT Procedure

Print ID SMOKE and AGE variables by RACE

```
----- race=A -----  
  
          id      smoke      Age  
  
      747F      current      18  
      796F      past        35  
      745F      never       36  
  
----- race=B -----  
  
          id      smoke      Age  
  
      135F      current      29  
  
...
```

The PRINT Procedure

```
----- race=H -----
      id      smoke      Age
      629F     past       26
      713F     never       26

----- race=W -----
      id      smoke      Age
      656F     never       26
      711F     never       32
      733F     current     17
      982F     past        26
      798F     never       19
      494F     never       36
      748F     never       34
      904F     never       25
      244F     never       28
      184M     past        19
```

The MEANS Procedure

```
PROC MEANS <DATA=SAS-data-set>  
          <MAXDEC=>  
          <statistic-keyword(s) >;  
  BY <DESCENDING> variable-1  
    <...<DESCENDING> variable-n>;  
  CLASS variable(s);  
  VAR variable(s);  
  WHERE where-expression;  
RUN;
```


The MEANS Procedure

Program 1.8

```
title 'The Mean Procedure - Class Statement';  
proc means data=hearing min median max maxdec=2 ;  
    where race = 'W' or race = 'B';  
    class smoke;  
    var age;  
run;
```

The Mean Procedure - Class Statement

The MEANS Procedure
Analysis Variable : Age

smoke	N Obs	Minimum	Median	Maximum
current	6	17.00	28.50	33.00
never	14	19.00	27.00	36.00
past	5	19.00	21.00	34.00

The MEANS Procedure

Program 1.8

```
proc sort data=hearing;  
    by smoke;  
run;  
title 'The Mean Procedure - By Statement';  
proc means data=hearing min median max maxdec=2;  
    where race = 'W' or race = 'B';  
    by smoke;  
    var age;  
run;
```

The Mean Procedure - By Statement

----- smoke=current -----

The MEANS Procedure

Analysis Variable : Age

Minimum	Median	Maximum
17.00	28.50	33.00

The MEANS Procedure

Program 1.8

```
----- smoke=never -----  
  
      Analysis Variable : Age  
  
      Minimum           Median           Maximum  
-----  
      19.00             27.00             36.00  
-----  
  
----- smoke=past -----  
  
      Analysis Variable : Age  
  
      Minimum           Median           Maximum  
-----  
      19.00             21.00             34.00  
-----
```

The FREQ Procedure

```
PROC FREQ <DATA=SAS-data-set> ;  
  BY <DESCENDING> variable-1  
    <...<DESCENDING> variable-n>;  
  WHERE where-expression;  
  TABLES requests </ options>;  
RUN;
```

- ❖ REQUESTS: specify frequency and contingency tables:
 - ❑ one-way frequency table: use one variable name
 - ❑ two-way contingency: write an asterisk between two variables

The FREQ Procedure

```
PROC FREQ <DATA=SAS-data-set> ;  
  BY <DESCENDING> variable-1  
    <...<DESCENDING> variable-n>;  
  WHERE where-expression;  
  TABLES requests </ options>;  
RUN;
```

- ☐ LIST: n-way tables are created in list format
- ☐ NOFREQ: suppresses frequencies
- ☐ NOCUM: suppresses cum. freq. and percentages
- ☐ NOPERCENT: suppresses percentages
- ☐ NOCOL: suppresses column percentages
- ☐ NOROW: suppresses row percentages
- ☐ MISSING: treats missing values as a group
- ☐ MISSPRINT: displays missing value frequency without percentages

The FREQ Procedure

Program 1.9

```
title 'No option';  
proc freq data=hearing;  
    tables preg;  
run;
```

No option				
The FREQ Procedure				
Preg	Frequency	Percent	Cumulative Frequency	Cumulative Percent

0	19	63.33	19	63.33
1	11	36.67	30	100.00

Frequency Missing = 4

The FREQ Procedure

Program 1.9

```
title 'Using MISSING option';  
proc freq data=hearing;  
    tables preg/missing;  
run;
```

Using MISSING option

The FREQ Procedure

Preg	Frequency	Percent	Cumulative Frequency	Cumulative Percent

.	4	11.76	4	11.76
0	19	55.88	23	67.65
1	11	32.35	34	100.00

The FREQ Procedure

Program 1.9

```
title 'Using MISSPRINT option';  
proc freq data=hearing;  
    tables preg/missprint;  
run;
```

Using MISSPRINT option

The FREQ Procedure

Preg	Frequency	Percent	Cumulative Frequency	Cumulative Percent

.	4	.	.	.
0	19	63.33	19	63.33
1	11	36.67	30	100.00

Frequency Missing = 4

The FREQ Procedure

Program 1.10

```
title 'Without LIST option';  
proc freq data=hearing;  
    tables preg*race*smoke;  
run;
```

The FREQ Procedure

Without LIST option
The FREQ Procedure
Table 1 of race by smoke
Controlling for Preg=0

race	smoke			
Frequency				
Percent				
Row Pct ,				
Col Pct	current	never	past	Total
-----+-----+-----+-----+				
A	2	1	1	4
	10.53	5.26	5.26	21.05
	50.00	25.00	25.00	
	33.33	11.11	25.00	
-----+-----+-----+-----+				
B	1	2	0	3
	5.26	10.53	0.00	15.79
	33.33	66.67	0.00	
	16.67	22.22	0.00	
-----+-----+-----+-----+				
H	0	1	1	2
	0.00	5.26	5.26	10.53
	0.00	50.00	50.00	
	0.00	11.11	25.00	
-----+-----+-----+-----+				
W	3	5	2	10
	15.79	26.32	10.53	52.63
	30.00	50.00	20.00	
	50.00	55.56	50.00	
-----+-----+-----+-----+				
Total	6	9	4	19
	31.58	47.37	21.05	100.00

The FREQ Procedure

Without LIST option
The FREQ Procedure
Table 2 of race by smoke
Controlling for Preg=1

race	smoke			
Frequency				
Percent				
Row Pct				
Col Pct	current	never	past	Total
-----+-----+-----+-----+				
A	0	1	0	1
	0.00	10.00	0.00	10.00
	0.00	100.00	0.00	
	0.00	12.50	0.00	
-----+-----+-----+-----+				
B	1	0	0	1
	10.00	0.00	0.00	10.00
	100.00	0.00	0.00	
	100.00	0.00	0.00	
-----+-----+-----+-----+				
H	0	1	0	1
	0.00	10.00	0.00	10.00
	0.00	100.00	0.00	
	0.00	12.50	0.00	
-----+-----+-----+-----+				
W	0	6	1	7
	0.00	60.00	10.00	70.00
	0.00	85.71	14.29	
	0.00	75.00	100.00	
-----+-----+-----+-----+				
Total	1	8	1	10
	10.00	80.00	10.00	100.00

Frequency Missing = 1

The FREQ Procedure

Program 1.10

```
title 'With LIST option';  
proc freq data=hearing;  
    tables preg*race*smoke/list;  
run;
```

The FREQ Procedure

With LIST option

The FREQ Procedure

Preg	race	smoke	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	A	current	2	6.90	2	6.90
0	A	never	1	3.45	3	10.34
0	A	past	1	3.45	4	13.79
0	B	current	1	3.45	5	17.24
0	B	never	2	6.90	7	24.14
0	H	never	1	3.45	8	27.59
0	H	past	1	3.45	9	31.03
0	W	current	3	10.34	12	41.38
0	W	never	5	17.24	17	58.62
0	W	past	2	6.90	19	65.52
1	A	never	1	3.45	20	68.97
1	B	current	1	3.45	21	72.41
1	H	never	1	3.45	22	75.86
1	W	never	6	20.69	28	96.55
1	W	past	1	3.45	29	100.00

Frequency Missing = 5

Subsetting Data by Selecting Variables

- ❖ Subsetting data: selecting a portion of a SAS data set by keeping only certain variables or selected observations (or both).
- ❖ When you subset data by selecting variables, you often need to keep or drop a list of variables.

Subsetting Data by Selecting Variables

❖ SAS *variable list* notation:

Variable list	Example	Equivalence
Numbered range lists	VAR1-VAR3	VAR1 , VAR2 , VAR3 or VAR1 VAR2 VAR3
Named range lists	ID--PREG	All the variables in order of variable creation from ID to PREG
	ID -NUMERIC- PREG	All numeric variables from ID to PREG
	ID -CHARACTER- PREG	All character variables from ID to PREG
Name prefix lists	DRUG:	All the variables that begin with “DRUG”
Special SAS name lists	<u>NUMERIC</u>	All numeric variables that are already defined in the current DATA step.
	<u>CHARACTER</u>	All character variables
	<u>ALL</u>	All variables

Selecting Variables with the KEEP= Data Set Option or KEEP Statement

- ❖ Data set options: specify certain actions that apply to either the input or output data sets.

(option-1=value-1<...option-n=value-n>)

- ❖ The KEEP= data set option:

KEEP=variable-list

KEEP=variable-1 <...variable-n>

- ❖ The KEEP= data set option can be used to apply to either the input and output data sets

Selecting Variables with the KEEP= Data Set Option or KEEP Statement

Program 1.11

```
data dat1;  
    set hearing(keep = id smoke age) ;  
run;
```

- ❖ Using the KEEP= option after DAT1 in the DATA statement will yield the same result
- ❖ Specifying the KEEP= option in the SET statement is more efficient because SAS only reads the desired variables.

Selecting Variables with the KEEP= Data Set Option or KEEP Statement

Program 1.11

```
data dat1;  
    set hearing(keep = id smoke age) ;  
run;
```

Log from Program 1.11

```
136  data dat1(keep = id smoke age);  
137      set hearing;  
138  run;
```

NOTE: There were 34 observations read from the data set
WORK.HEARING.

NOTE: The data set WORK.DAT1 has 34 observations and 3
variables.

NOTE: DATA statement used (Total process time):

real time	0.07 seconds
cpu time	0.01 seconds

Selecting Variables with the KEEP= Data Set Option or KEEP Statement

❖ The KEEP statement:

KEEP variable-list;

KEEP variable-1 <...variable-n>;

Program 1.12

```
data dat1;  
    set hearing;  
    keep id smoke age;  
run;
```

Selecting Variables with the DROP= Data Set Option or DROP Statement

❖ The DROP= data set option:

DROP=variable-list

DROP=variable-1 <...variable-n>

❖ The DROP statement:

DROP variable-list;

DROP variable-1 <...variable-n>;

Selecting Variables with the DROP= Data Set Option or DROP Statement

Program 1.13

```
data dat2;  
    set hearing (drop= race preg -- income);  
run;
```

Program 1.14

```
data dat2;  
    set hearing;  
    drop race preg -- income;  
run;
```

- ❖ If you have more variables to drop than you have to keep, it will be easier for you to use the KEEP= option or the KEEP statement

Where to Specify the DROP= and KEEP= Data Set Options and DROP/KEEP Statements

Program 1.15

```
data dat3 (drop= income);  
    set hearing (keep= id income);  
    if income > 50000 then income_hi = 1;  
    else income_hi = 0;  
run;
```

Where to Specify the DROP= and KEEP= Data Set Options and DROP/KEEP Statements

- ❖ The DROP= and KEEP= options can be used to apply to either the input and output data sets.
- ❖ However, the KEEP and DROP statements apply only to output data sets.

Program 1.16

```
data dat4 (keep= id race smoke)
      dat5 (keep= id age preg) ;
  set hearing;
run;
```

Changing the Appearance of Data

- ❖ Labeling variable names

- ❖ Formatting variable

Labeling Variables

❖ The LABEL statement:

```
LABEL variable-1=label-1 . . .  
    <variable-n=label-n>;
```

- ❖ The labels can contain blanks and ≤ 256 characters.
- ❖ If the label contains semicolons (;) or equal signs (=), you can enclose the label in either single or double quotations.
- ❖ If the label contains a single quote ('), you must enclose the labels in double quotations.

Labeling Variables

❖ To remove labels from variables:

```
LABEL variable-1=' ' ... <variable-n=' '>;
```

Labeling Variables

- ❖ The LABEL statement can be used in either the DATA step or PROC step.
- ❖ When using a LABEL statement in a DATA step, you associate labels with the variables permanently
- ❖ When using a LABEL statement in the PROC step, the assigned label is only available to the output that the current procedure generated.

Labeling Variables

Program 1.17

```
data hearing1_1;  
  set hearing;  
  label hearing = Hearing Loss  
  income = "People's Income";  
run;  
  
title 'Assigning Labels Permanently';  
proc print data=hearing1_1(obs=5) label ;  
  var hearing income;  
run;
```

Assigning Labels Permanently		
	Hearing	People's
Obs	Loss	Income
1	no	29000
2		28700
3		59000
4	no	120000
5		29000

Labeling Variables

Program 1.17

```
proc contents data=hearing1_1;  
run;
```

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Label
4	Age	Num	8	
6	Hearing	Char	3	Hearing Loss
7	Income	Num	8	People's Income
5	Preg	Num	8	
1	id	Char	4	
2	race	Char	1	
3	smoke	Char	7	

Labeling Variables

Program 1.18

```
title 'Assign temporary label to RACE variable';  
proc means data=hearing1_1 mean median std;  
    label race = Ethnicity;  
    class race;  
    var income;  
run;
```

Assign temporary label to RACE variable

The MEANS Procedure

Analysis Variable : Income People's Income

N				
Ethnicity	Obs	Mean	Median	Std Dev
A	5	61420.00	39100.00	45499.25
B	5	61200.00	39100.00	44726.45
H	4	44800.00	37000.00	21332.92
W	20	53575.00	48550.00	25353.54

Formatting Variable Values Using SAS FORMATS

- ❖ Format: an instruction to tell SAS how to write data values in the output.
- ❖ Variable values can be formatted by using either SAS formats or user-defined formats
- ❖ The FORMAT statement:

```
FORMAT variable-1 <. . . variable-n> format  
variable-1 <. . . variable-n> format;
```

Formatting Variable Values Using SAS FORMATS

❖ Formatting standard character data:

`$w.`

- ☐ \$ is required for formatting character values.
- ☐ w: total width of the character values
- ☐ The period (.) is a required

Formatting Variable Values Using SAS FORMATS

- ❖ The non-standard character formats contain a keyword in between \$ and the w field.

`$UPCASEw.`

`$QUOTEw.`

Formatting Variable Values Using SAS FORMATS

❖ Formatting standard numeric values:

w.d

- ❑ w: total width of the numerical values, including the decimal point.
- ❑ The second field is a period (.)
- ❑ d: the # of digits to the right of the decimal point.
- ❑ If d is omitted, w.d format writes the value without a decimal point.

Formatting Variable Values Using SAS FORMATS

- ❖ A non-standard numeric format contains a keyword in front of the w field.

DOLLARw.d

- ☐ a leading dollar sign in the starting position
- ☐ a comma that separates every three digits
- ☐ a period that separates the decimal fraction
- ☐ w must be large enough

Formatting Variable Values Using SAS FORMATS

❖ To disassociate a format from a variable:

```
FORMAT variable-1 <. . . variable-n>;
```

Formatting Variable Values Using SAS FORMATS

Program 1.19

```
data hearing1_2;  
    set hearing;  
    format smoke hearing $upcase5. income dollar11.2;  
run;  
  
title 'Assigning formats to variable';  
proc print data=hearing1_2(firstobs=6 obs=12);  
    var smoke hearing income;  
run;
```

Assigning formats to variable

Obs	smoke	Hearing	Income
6	CURRE		\$19,000.00
7	CURRE		\$23,900.00
8	CURRE		\$39,000.00
9	CURRE		\$39,100.00
10	NEVER	NO	\$48,000.00
11	NEVER	NO	\$30,000.00
12	NEVER		\$25,000.00