



Intermediate SAS

Course Objective

This course would help students to understand the structure and design of the SAS system and write program in SAS syntax

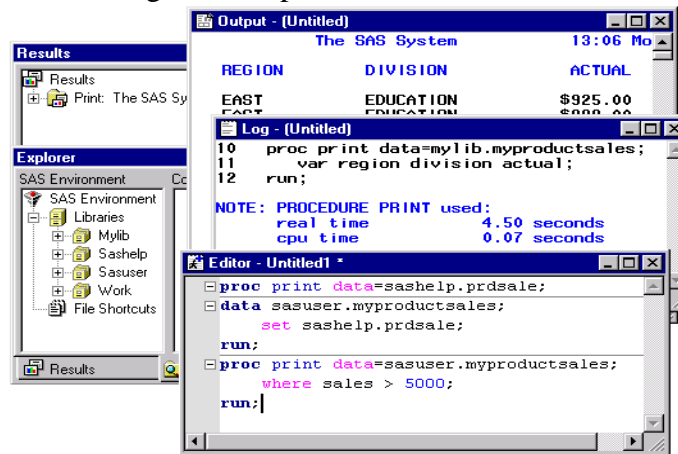
Four data-driven tasks of SAS

The functionality of the SAS System is built around the four data-driven tasks common to virtually any application:

- Data access
- Data management
- Data analysis
- Data presentation

SAS windowing environment

- Five Important Windows
- Explorer, Results, Editor, Log, and Output



SAS Syntax rules

- SAS statements usually begin indentifying keywords, always end with a semicolon (;)
- We enter this code in the Editor window
- SAS statements are free format
- You can type notes (comments) within the code by simply putting an asterisk (*) in front of your note

The Three Most Common Errors

- Leaving out a semi-colon
- Misspellings
- Leaving out a quote (‘) Quotes make sure SAS reads the characters between the quotes as a character string

SAS programs

A *SAS program* is a sequence of steps that the user submits for execution.

Data steps – typically used create a SAS data sets

Proc steps – typically used to process SAS data sets

The End – Always end with the following **RUN;**

Submitting a SAS programs

When you execute a SAS program, the output generated by SAS is divided into two major parts:

- SAS log -- contains information about the processing of the SAS program, including any warning and error messages.
- SAS output -- contains reports generated by SAS procedures and DATA steps.

SAS data library

When we invoke SAS, we automatically have access to temporary and permanent SAS data library

- work- temporary library
- sasuser- permanent library
- ia- permanent library

You can also create and access your own library

`libname` statement assigns a libref to SAS data library, general form of the libname statement:

```
libname libref 'SAS-data-library' <options>;
```

Import data to SAS Season

There are few ways to import data to SAS depending on your data set.

- From Raw data
- From External files(excel file and text file)
- From other software files

Reading raw data files

- First we must give our data set a name....

```
data SAS-data-set-name;
```

- Now that we have a name we must give it some variables that we want to use....

```
input variable <$> startcol-endcol...;
```

-the character variables need a \$ placed after them

-variables must be between 1-8 characters and begin with a letter

- After we have told SAS what variables we want to use, we must enter in our data line by line...

Cards;

```
43912/11/00LAX 20137
92112/11/00DFW 20131
11412/12/00LAX 15170
98212/12/00dfw 5 85
43912/13/00LAX 14196
98212/13/00DFW 15116
43112/14/00LaX 17166
98212/14/00DFW 7 88
11412/15/00LAX 187
98212/15/00DFW 14 31
```

- And we end with...

Run;

```
data SAS-data-set-name;;
  infile 'row-data-file';
  *Tells our data set where to look
  input input-specifications;
  *Tells SAS the names of our variables
run;
```

Reading External and other software files

```
PROC IMPORT OUT= WORK. SAS-data-set-name
  DATAFILE= "G:\\SAS-data-set.txt"
  DBMS=TAB REPLACE;
  GETNAMES=YES;
  DATAROW=2;
RUN;
```

SAS Procedures

SAS software provides a variety of procedures that produce reports, compute statistics, and perform utility operations.

Proc Print Data=cars;

This procedure will print all the data that we inputted from before! (Note: Check the output handout to see if you did this correctly.)

Proc SORT Data=cars;
By year;

This procedure will sort our data by year number.

```
Proc Means Data=cars <Options>;
  Var mpg horse accel;
  Output out=carsMEANS;
```

This procedure will produce output of basic descriptive statistics for each variable that you specify.

```
Proc Freq Data=cars;  
  Tables mpg accel/<Options>;  
  Output out=carsMEANS;
```

This procedure creates cross tabulation tables and can measure association with <Options>
Some options are *chisq nocol norow nopercnt*

```
Proc Corr Data=cars;  
  Var mpg;  
  With horse;
```

This procedure analyzes the correlation between variables (mpg and horse in this case)

```
Proc Univariate Data=cars <Options>;  
  Var mpg;  
  By year;  
  Output Out=carsUNI;
```

This procedure will produce basic statistics for each variable that you specify on the VAR line.

```
Proc ANOVA Data=cars <Options>;  
  Class horse accel;  
  Model mpg=horse accel;  
  Output Out=carsANOVA;
```

This procedure will produce an Analysis Of Variance for the independent variables ‘horse’ and ‘accel’ and the dependent variable ‘mpg.’

```
Proc Reg Data=cars <Options>;  
  Model mpg= horse engine / <Options>;  
  Plot mpg*engine;
```

This procedure will do a simple regression analysis where ‘horse’ and ‘engine’ are our independent variables and mpg is the dependent variable. The plot statement will plot the regression line for the above model.

```
Proc Plot Data=cars;  
  Plot mpg*horse;  
  Output Out=carsPLOT;
```

This procedure will produce a scatter plot with mpg vs. horse.

```
Proc Chart Data=cars <Options>;  
  Pie year;
```