การอบรม

เรื่อง

"Introduction to SAS Programming"

โดย

รองศาสตราจารย์ ดร.พันทิศา ภาวบุตร

วันพุธที่ 25 เมษายน 2550
เวลา 09:00 – 10:30 น.
ศูนย์คอมพิวเตอร์ ชั้น 2

ศูนย์วิจัยธุรกิจ
คณะพาณิชยศาสตร์และการบัญชี
มหาวิทยาลัยธรรมศาสตร์

THAMMASAT
BUSINESS SCHOOL

Why are we here?

- To share our experience in using SAS with you
- Tell you why we like using SAS



Class Plan: April 25, 2007

Morning Session:

- Overview of SAS Software Products
- Getting started using SAS
- Getting your data into the SAS System

Afternoon session:

- Working with data
- Simple statistics

What is SAS?

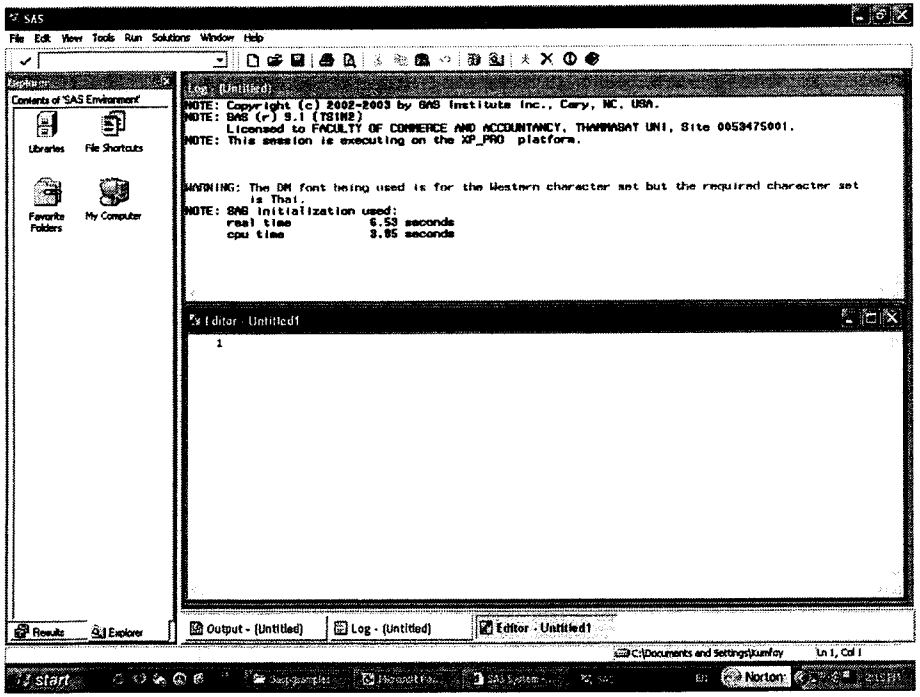
- *The **SAS System***, originally **S**tatistical **A**nalysis **S**ystem, is an integrated system of software products provided by SAS Institute that enables the programmer to perform:
- data entry, retrieval, management, and mining
- report writing and graphics
- statistical and mathematical analysis
- business planning, forecasting, and decision support
- operations research and project management
- quality improvement
- applications development
- data warehousing (extract, transform, load)
- platform independent and remote computing

SAS Interface

- SAS for programmers
- SAS for non-programmers
  - SAS Enterprise
  - SAS/Assist, Analyst

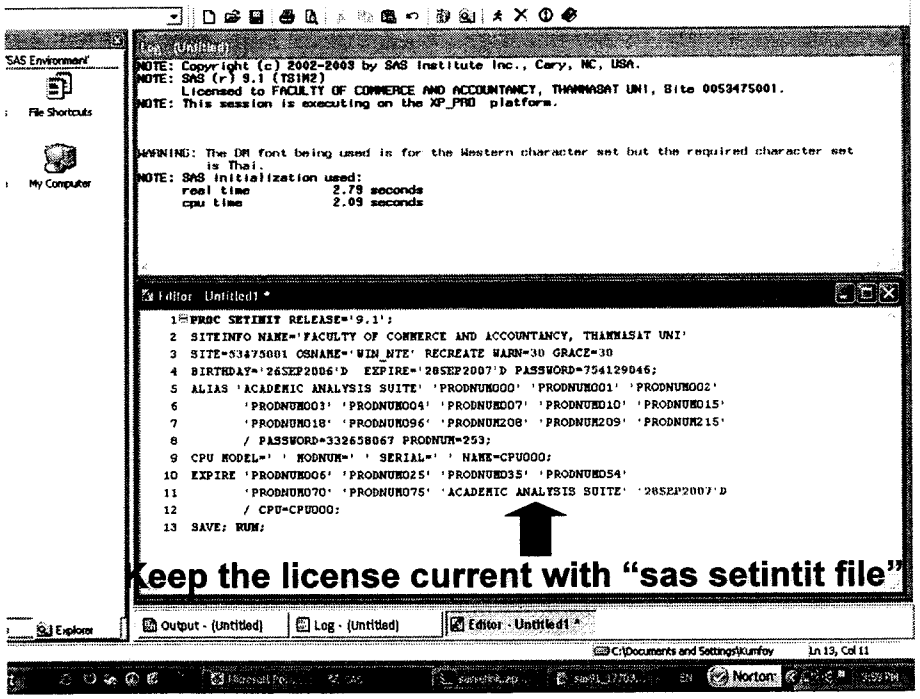
SAS Components

- Base SAS: SQL and macro facility
- SAS Enterprise
- SAS/ETS: Econometrics and time series
- SAS/IML: Interactive matrix language
- SAS/OR: Optimization tools
- SAS/STAT: Statistical analysis, ANOVA and simple regressions



Getting help with SAS

- **Useful references (Beginners)**
  - The little SAS book: L. Delwiche and S. Slaughter
  - SAS programming by example: Ron Cody and Ray Pass
- **On-line help**
  - Google for the code
  - SAS on-line document in pdf from http://www.sas.com/
  - Check out sample codes on
  - http://support.sas.com/rnd/app/examples/index.html
- **SAS Support Help desk**
  - Email: support@sas.com
  - name=Sopa Ngarm
    site= 0053475001
    company=Thammasat University.
    phone=+66 (2) 225-2107
    product=Base SAS
    release=9.1
    os=MS Windows



Getting started with SAS

- Creating and submitting a SAS program
- Recalling a program
- Error messages

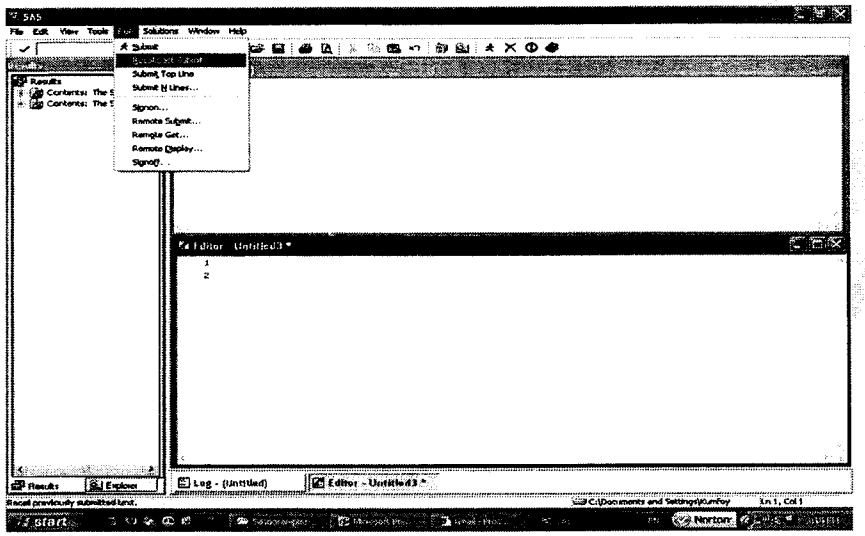
Creating and submitting a program

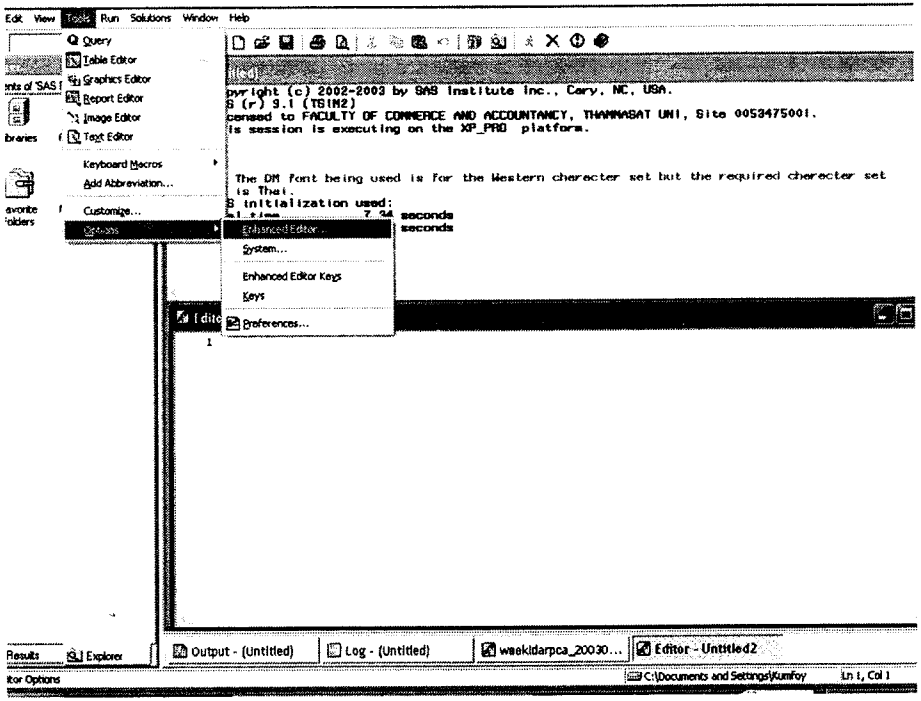
```
proc import datafile='D:\saspgsamples\sasex1.xls'

out = ex1 ;
format date MMDDYY8. ;

run ;


proc contents data=ex1 ;
run;
```

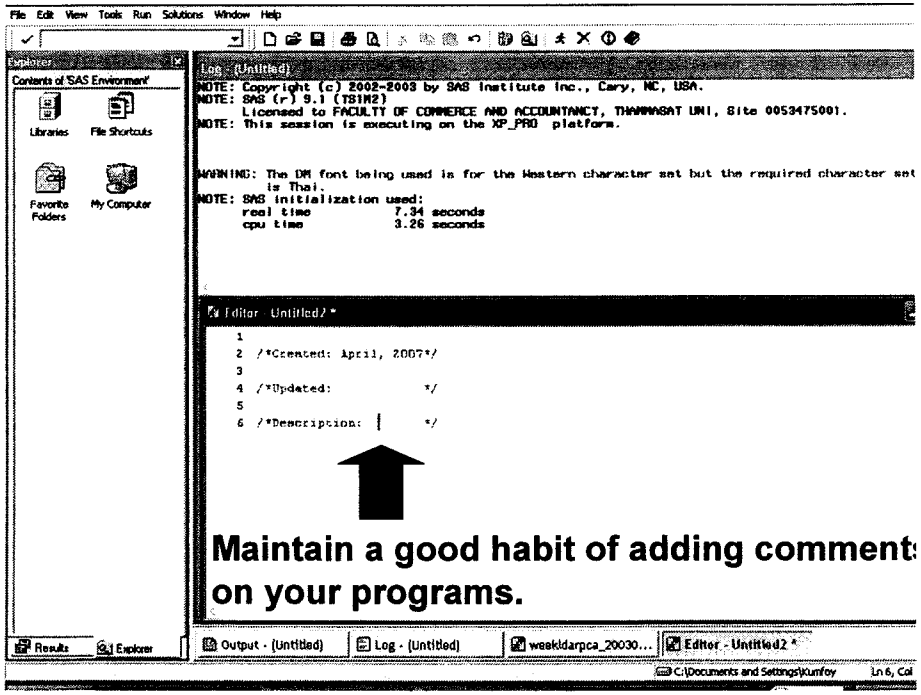

Windowing environment

- F5 for PGM window
- F6 for log window
- F7 for output window

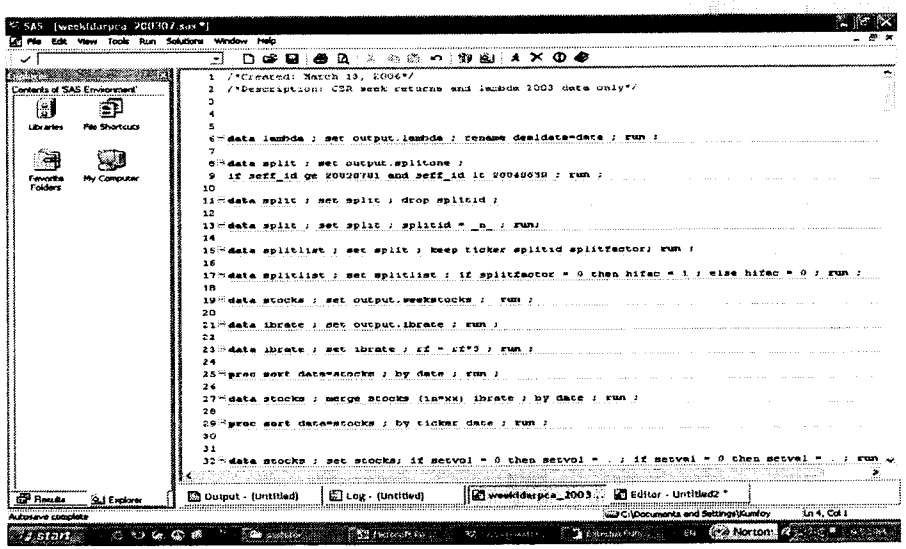
Recall a program

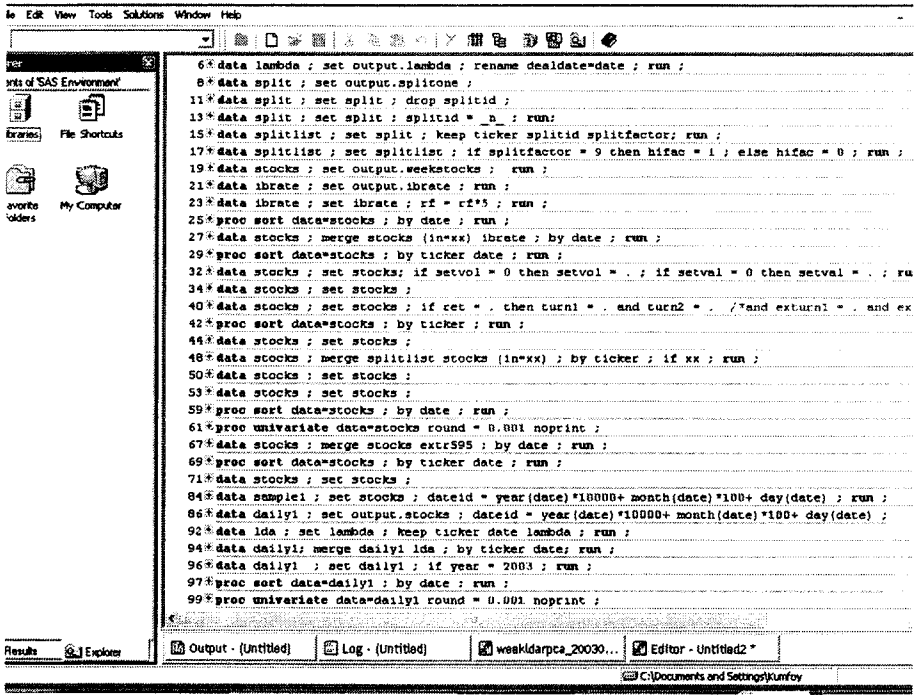






To see more of your program condensed, try "CTRL, ALT, -"





SAS/STAT

```
data ex1 ; set ex1 ; total = part_1 + part_2 +
  part_3 ; run ;

proc univariate data=ex1 ;
var total ;
output out= score mean=avgsc
  median=medsc max=maxsc min=minsc
  q1=q1sc q3=q3sc ;
run;
```


SAS/OR

```
proc nlp ;
min f ;
decvar x1 x2 ;
f1 = 10*(x2-x1*x1) ;
f2 = 1- x1 ;
f = 0.5 * (f1 * f1 + f2 * f2) ; run ;
```


SAS/IML

```
proc iml ;
a={1,1,1} ;
b= 2*a ;
print b ; run ; quit;

proc iml;
use ex1 ;
   read all var {part_1 part_2 part_3} into X ;
   tx = t(x) ;
   print tx ;
   run ; quit ;
```

MATLAB vs SAS

Matlab

- Complex problem involving matrix algebra
- High quality chart
- Specialized applications available, ie. Technical analysis
- Poor data manipulation and handling
- Better for simulation
- Cheaper
- Overall: Great for solving maths problem (with numerical solutions)

SAS

- Solving algebra can be done via SAS/OR
- Charting is cumbersome
- Best program existing for data handling
- Expensive
- Manual difficult to use
- Five-star technical support
- Overall: Great for statistical use from simple to very complex.

SPSS vs SAS

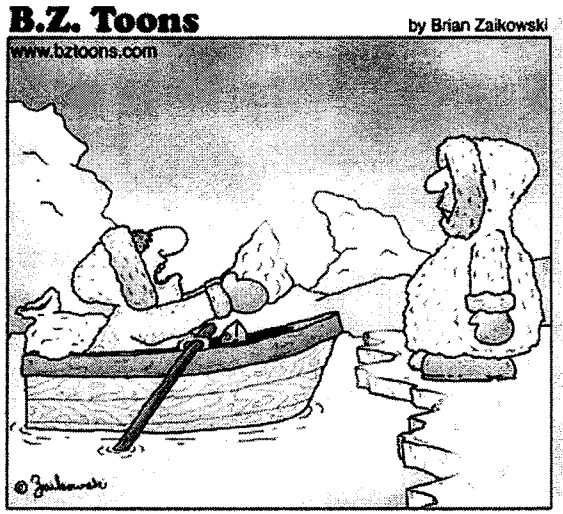
SPSS

- Easier to use
- Poor handling of large datasets.
- Slow processing speed
- Cheaper

SAS

- More difficult to use
- Powerful and fast for handling complex problems with large datasets.
- More advanced econometrics tool available

SAS
The Power to Know



I got you a bunch of stuff. This is just the tip of the icegerg.

การอบรม

เรื่อง

"Introduction to SAS Programming"

โดย

อาจารย์ ดร.ปิยรัตน์ จันทะเดช

วันพุธที่ 25 เมษายน 2550
เวลา 10:30 – 16:00 น.
ศูนย์คอมพิวเตอร์ ชั้น 2

ศูนย์วิจัยธุรกิจ
คณะพาณิชยศาสตร์และการบัญชี
มหาวิทยาลัยธรรมศาสตร์

Introduction to SAS Programming

April 25-26, 2007
Piyaratt Jantadej, Ph.D.

This document covers the following topics:-

- Getting started using SAS software
- Getting data into the SAS system
- Modifying/Combining SAS data sets
- Working with SAS data
- Using basic statistics

Section 1: Getting Started Using SAS Software

1.1) The SAS Language

The SAS system helps you to organize and analyze a collection of data items using SAS programming statements. A SAS program is collection of SAS statements in a logical sequence. There are two major components of a SAS program: **DATA steps and PROC steps** (or PROCedure steps). Typically, a program starts with DATA steps to create a SAS data set and then passes the data to PROC steps for processing. Specifically, DATA steps read and modify data while PROC steps analyze data, perform utility function, or print reports.

DATA steps	PROC steps
Begin with DATA statements	Begin with PROC statements
Read and modify data	Perform specific analysis or function
Create a SAS data set	Produce results or reports

The following sample is a simple program that converts miles to kilometers in a DATA step and produces a SAS data set named distance, and then prints the results with a PROC step. Note that a RUN statement tells SAS to run all the preceding lines.

```
DATA distance;
      Miles = 30;
      Kilometers = 1.61 * Miles;

PROC PRINT DATA= distance;
      Run;
```

DATA step

PROC step

The most important rule about how to **format** your SAS program is every SAS statement ends with *a semicolon.* SAS statements can be in *upper- or lowercase.* Almost SAS statements begin with a SAS keyword (e.g., data, set, proc, title, if, etc.). Exceptions are assignment statements (e.g., commission fees = sales * 0.15).

You can insert **comments** into your programs to make programs more understandable. There are two styles of comments: one starts with an asterisk (*) and ends with a semicolon (;). The other style starts with a slash asterisk (/*) and ends with an asterisk slash (*/).

Examples of comments

```
* Create a data set named distance for converting miles to kilometers;

      DATA distance;
            Miles = 30;
            Kilometers = 1.61 * Miles;

/* Print the results*/

      PROC PRINT DATA= distance;
            Run;
```

SAS program

1.2) SAS Data Sets

Before SAS can read your data, the data must be in a form called a SAS data set. The data consists of **variables and observations**.

The following table contains a sample of a small data set. Each line represents one observation, while Id, Name, Height, and Weights are variables.

Variables (also called Columns)

Observations (also called Rows)

	Id	Name	Height	Weight
1	23	Thanat	167	57
2	24	Suree	155	45
3	25	Malee	162	.
4	26	Ampawa	157	50
5	27	Surapong	180	70

1.2.1 Data Types

There are two data types in a SAS data set: **numeric and character**. **Numeric fields** are numbers, and they can be added and subtracted, can have any number of decimal places, and can be positive or negative. Also, numeric fields can contain plus sign (+), minus sign (-), or E for scientific notation. **Character data** are everything else. They may contain numerals, letters, or special characters (such as $).

1.2.2. Size of SAS Data Sets

SAS can handle up to 32,767 variables in a single data set. The number of observations is limited only by your computer's capacity to handle and store them.

1.2.3 Rules for SAS Names

Rules for **variable names and data set member names** are: (1) names must be 32 characters or fewer in length; (2) names must start with a letter or an underscore (_); (3) names can contain only letters, numerals, or underscores; and (4) names can contain upper- and lowercase letters.

1.2.4 Temporary versus Permanent SAS Data Sets

A SAS program creates a SAS data set, either permanent or temporary, from raw data or from another SAS data set. **A temporary SAS data set** is one that exists only during the

current job or session and is automatically erased by SAS when you finish. This is often useful when you are in the testing stages of your analysis or when you do not want to keep a data set for later use. **A permanent SAS data set** is one that is created in the program and remains when the job or session is finished. Creating a permanent SAS data set is useful when you expect to work with the data set repeatedly.

How does SAS know you want a temporary data set or a permanent data set? SAS knows by the name you give the data set when you create it. **All SAS data sets have a two-level name**. The *first level* is called libref (i.e., SAS data **library** reference), and *the second level* is **the member name**. A libref is a nickname that refers to a location of a SAS data library (i.e., a directory that you want SAS data sets to be located), while a member name is just a name of data set. Note that librefs cannot be longer than 8 characters, whereas member names can be up to 32 characters long.

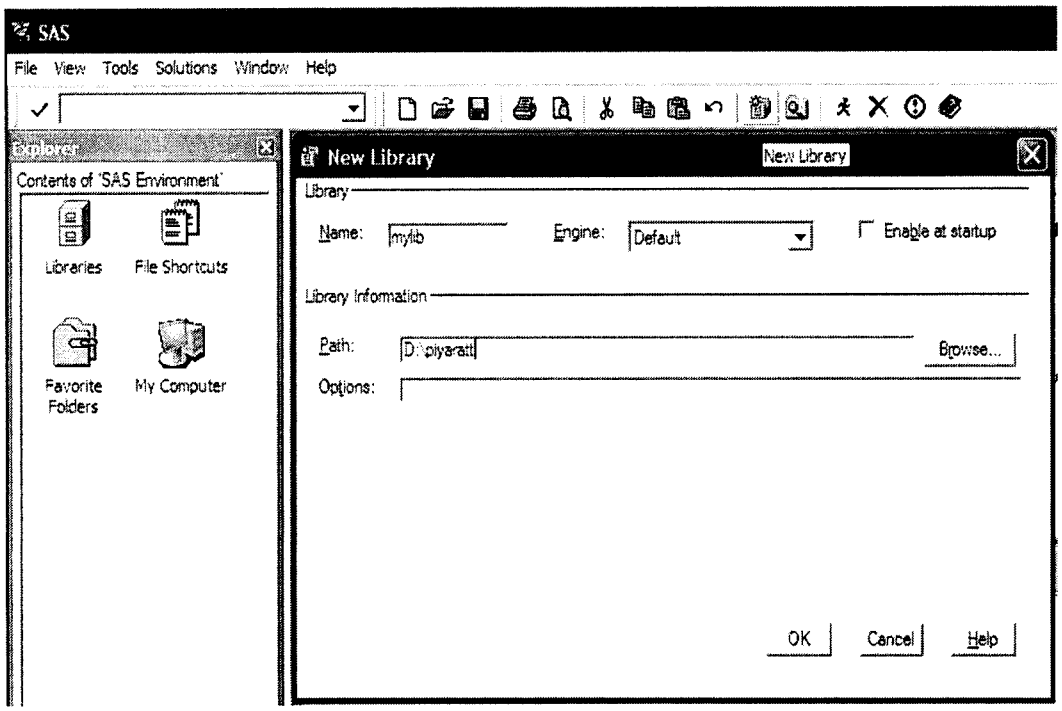
When you do not specify a libref of your own, SAS will put your data set in the default **WORK library**. Note that all data sets in the WORK library are erased by SAS at the end of the current job or session.

For example, I assign library name as 'mylib', and mylib library is located at 'D:\piyaratt'.

SAS Data Set Name	Libref	Member name	Type of Data Set
mylib.one	mylib	one	permanent
two	WORK	two	temporary

1.2.5 How to Create a Library Name?

Steps: 1) Go to Start Menu → Click Programs → Click SAS → Click SAS 9.1 (English)
2) Go to SAS Toolbar, Click New Library Icon
3) Type Library name 'mylib' and Path 'D:\piyaratt' as shown below, and then Click OK



Section 2: Getting Data into the SAS System

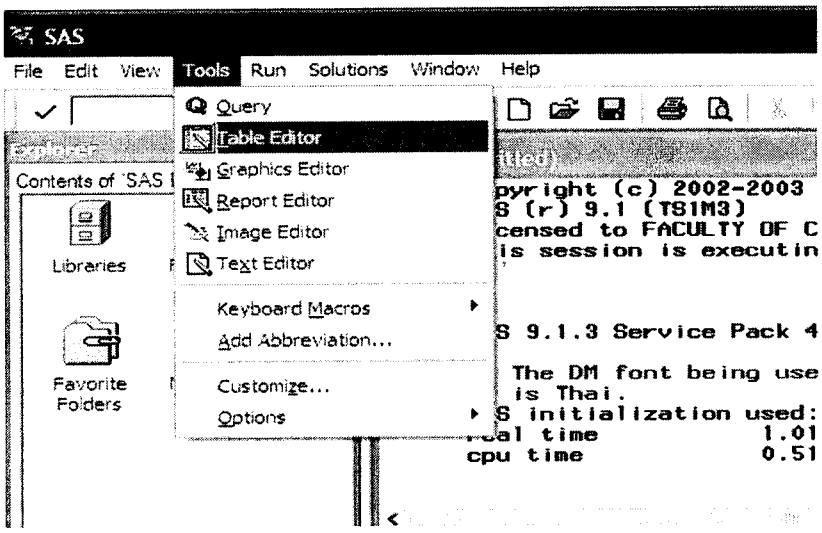
2.1) Methods for Getting Data into the SAS System

Before using SAS software, you need to put data into a SAS data set. Methods to get data into the SAS system include:-

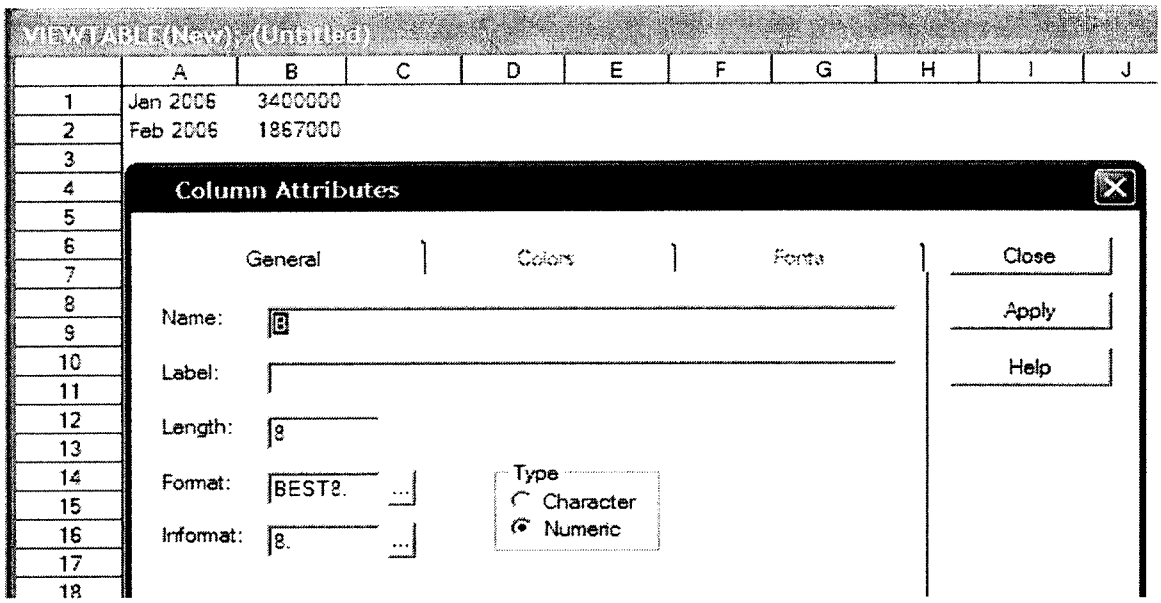
2.1.1 Entering data directly into SAS data sets

2.1.1.1. Using viewtable window

Steps: 1) Click the Tools Menu
2) Select Table Editor



3) Open a column attributes window to set up attributes for each column
4) Type data into this default table



2.1.1.2 Writing a SAS program and putting data in a DATA step

The INPUT and datalines statements are used as shown below.

Example:

```
* Generate a data set;

data mylib.productsale;
   input idno $ productname $ amount;
   datalines;
   T001 Table1 400
   T002 Table2 250
   C130 Chair13 780
   C160 Chair16 999
   B111 Bed11  25000
   B121 Bed12  0
   ;
   run;
```

2.1.2 Converting other software's data files (such as Excel Files [.xls extension], Comma Delimited Files [.CSV extension]) into SAS data sets

2.1.2.1 Using the IMPORT procedure (Proc Import)

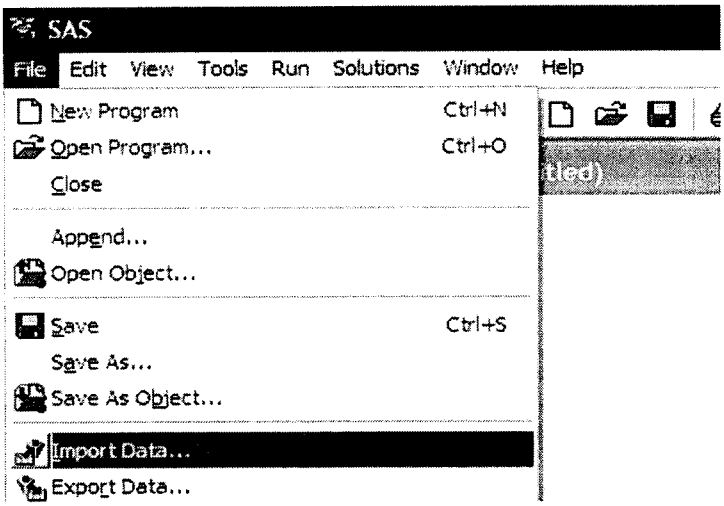
PROC IMPORT DATAFILE = 'filename' OUT = data-set REPLACE;

Example:

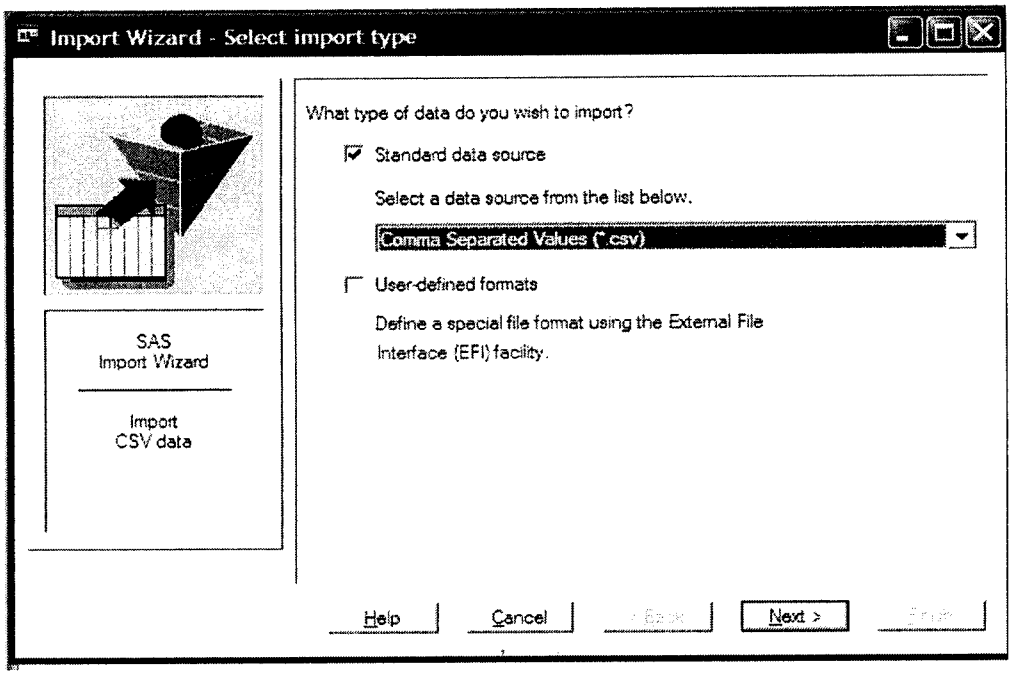
```
* Use proc import to import an external file into a SAS system;

proc import datafile = 'D:\piyaratt\rawdata\freq.csv' out = mylib.freq
replace;
run;
```

2.1.2.2 Using the IMPORT Wizard

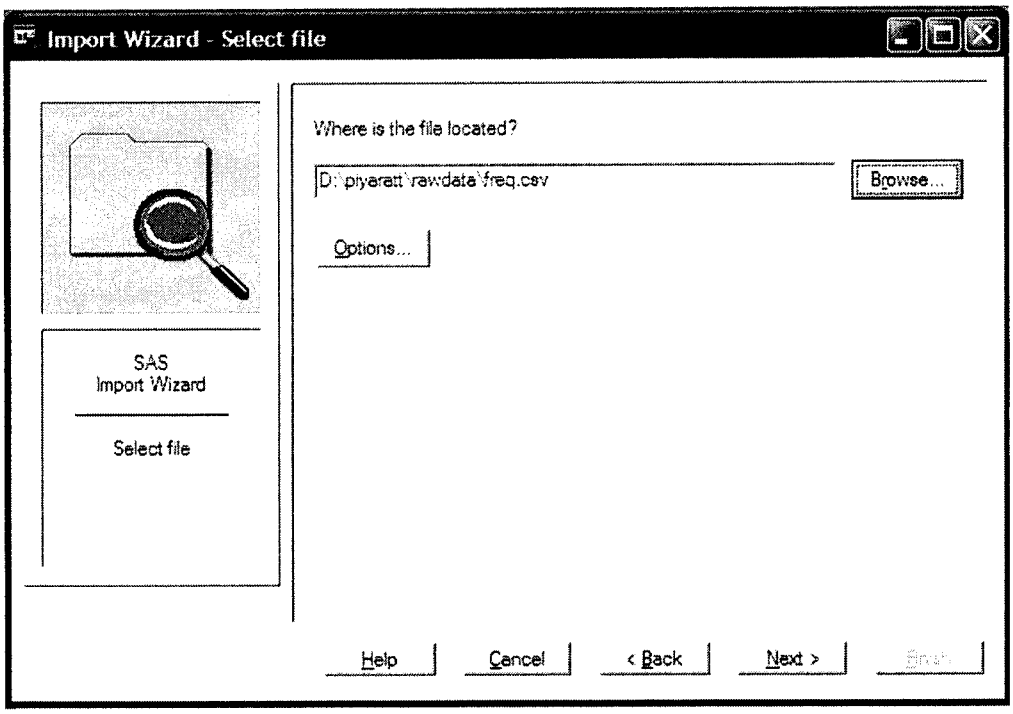
Steps: 1) Click the File Menu
2) Select Import Data



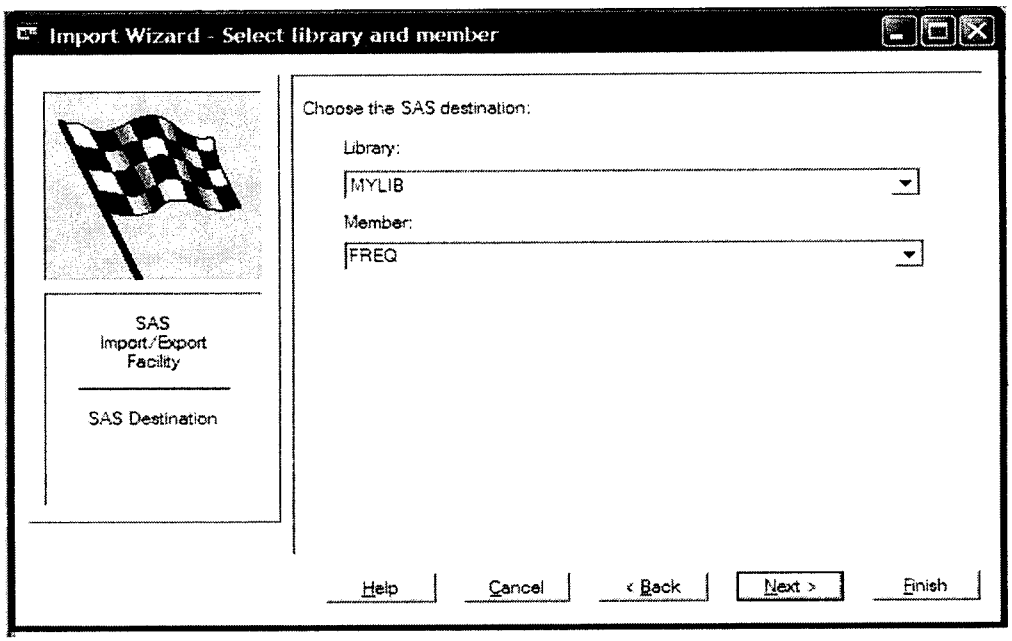
3) Select a data source as *.CSV, and then Click Next



4) Click Browse to search for a file you want to import (i.e., D:\piyaratt\rawdata\freq.csv), and then Click Next



5) Type a specified library name called 'mylib' and a member name called 'freq', and Click Finish



After steps 1-5 are performed, SAS creates a data set named 'mylib.freq', and this file is located at 'D:\piyaratt'.

2.2) Formatting the Raw Data

Raw data may not be available in formats you wanted. For instance, in one data set, you have a date variable in the form of DD/MM/YYYY, but you want to work with this variable in the form of MM/DD/YY instead because a date variable in other data sets is in the MM/DD/YY format. SAS can easily convert the date format for you with **the FORMAT statement**. The format statements can be written in either DATA steps or PROC steps.

FORMAT variable format;

An example of format statement is:

```
* import data from D:\piyaratt\rawdata\formatdate;
** format date from DD/MM/YYYY to MM/DD/YY;

proc import datafile ='D:\piyaratt\rawdata\formatdate.csv' out = mylib.formatdate;
   format date MMDDYY8.;
run;
```

The following table lists some of the date formats available in SAS software.

Format **Example**	**Description**	**Default *W*idth**
YYMMDD*w*. 99/04/25	Year, month,day:yy/mm/dd	8
MMDDYY*w*. 04/25/99	month, day, year: mm/dd/yy	8

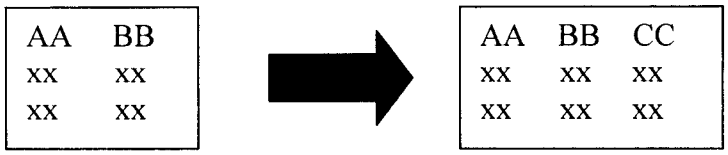
Format Example	Description	Default *Width*
MMYYC*w*. 04:1999	month and year: mm:yy	7
MMYYD*w*. 04-1999	month and year: mm-yy	7
MONTH*w*. 4	month of year	2
QTR*w*. 2	quarter of year	1
YEAR*w*. 1999	year	4
WEEKDAY*w*. 1	Day of week (1 = Sunday,...,7 = Saturday)	1

Also, the SAS system provides functions to perform calculations with SAS date, time, and datetime values. The following table lists some of SAS date functions.

Function Example	Description
YEAR (date) YEAR (04/25/99) = 1999	returns the year from a SAS date value
MONTH (date) MONTH (04/25/99) = 4	returns the month of the year from a SAS date value
QTR (date) QTR (04/25/99) = 1	returns the quarter of the year from a SAS date value
WEEKDAY (date) WEEKDAY (04/25/07) = 4	returns the day of the week from a SAS date value (Sunday = 1, ..., Saturday =7)

Section 3: Modifying and Combining SAS Data Sets

3.1) Modifying a Data Set Using the SET Statement



The set statement in the DATA step allows you to read a SAS data set so you can add new variables, create a subset, or modify the SAS dataset. The form of the DATA and SET statements is:

```
* SET statement;

    DATA new-data-set;
        SET old-data-set;
```

```
/* create new variable*/
        CC = AA + BB;
RUN;
```

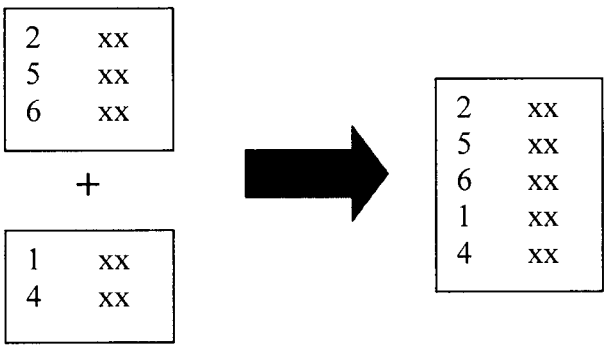
SAS reads data from old-data-set (i.e., AA and BB variables), then calculates a new variable named CC, and then keeps all three variables in new-data-set. *Why do you need to create new-data-set from these statements?* You want to keep an original data set as it is for later use so it is more efficient to work with new variables within other data sets. However, when you do not want to create the new data set, you can ask SAS to keep all variables (old and new) in the original data set.

```
DATA original data-set;
        SET original data-set;
        CC = AA + BB;
RUN;
```

An Example of the SET statement is:

```
* create new variable commission fee (5% of sale amounts for month # 2;
* and then keep all variables in other data set;

data mylib.freqfee;
       set mylib.freq;
       fee_mth2 = sales_mth2 * 0.05;
run;
```

3.2) Stacking Data Sets



When you want to combine data sets with all or most of the same variables but different observations, you can use **the SET Statement or PROC APPEND** to stack the data sets one on top of the other. The number of observations in the new data set will equal the sum of the number of observations in the old data sets. The order of the observations is determined by the order of the list of old data sets. If one of the data sets has a variable not contained in the other data sets, then the observations from the other data sets will have missing values for that variable.

3.2.1 Using the SET Statement in the DATA Step

In the DATA step, first specify the name of the new SAS data set in the DATA statement, and then list the names of the old data sets you want to combine in the SET statement.

```
DATA new-data-set;
        SET data-set-1 data-set-n;
```

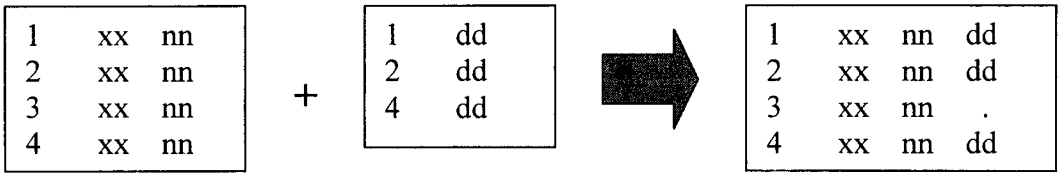
3.2.2 Using PROC APPEND in the PROC Step

```
PROC APPEND
        BASE = the larger data set or new-data-set
        DATA = the smaller data set-1;
RUN;
PROC APPEND
        BASE = the larger data set or new-data-set
        DATA = the smaller data set-2;
RUN;
.
.
.
PROC APPEND
        BASE = the larger data set or new-data-set
        DATA = the smaller data set-n;
RUN;
```

An example of PROC APPEND is:

```
* combine formatdateyr1990 and formatdateyr1991;
* create new data set called mylib.append2yr;

proc append
        base = mylib.append2yr
        data = mylib.formatdateyr1990;
run;

proc append
        base = mylib.append2yr
        data = mylib.formatdateyr1991;
        run;
```

3.3) Combining Data Sets Using the Merge Statement



When you have two SAS data sets with related data and you want to combine them, you can use the MERGE statement in the DATA step. In order to merge, you need to have any **common variable(s)** (or called the matching variable(s)) between these two data sets. Note that

the common variable(s) of these two data sets must be **in the same order** before you ask SAS to merge data sets for you!

The form of the MERGE statement within the DATA step is:

```
DATA new-data-set;
        SET data-set-1 data-set-2;
        BY variable1, ..., variable n;
```

The form of PROC SORT is:

```
PROC SORT;
        BY variable 1, ..., variable n;
```

An example of the merge statement and Proc sort is following.

```
* sort companytwo by idno;

data mylib.companytwo1;
        set mylib.companytwo;
proc sort;
        by idno;
run;

* merge 2 data sets by idno and year;

data mylib.company;
        merge mylib.companyone mylib.companytwo1;
        by idno year;
run;
```

3.4) Using SAS Data Set Options

Data set options affect how SAS reads or writes an individual data set. You can use data set options in DATA steps (in DATA, SET, or MERGE statements) or in PROC steps (with a DATA = statement option). To use a data set option, you only put it between parentheses directly following the data set name.

These are the most frequently used data set options.

KEEP= variable-list	tells SAS which variables to keep
DROP= variable-list	tells SAS which variables to drop
RENAME= (oldvar = newvar)	tells SAS to rename certain variables
FIRSTOBS= n	tells SAS to start reading at observation n
OBS = n	tells SAS to stop reading at observation n

Example:

```
* KEEP and RENAME function;
** keep date, nasdaq, and WD;
*** then rename WD ==> Weekday;

data mylib.nasdaq (rename = (WD = weekday));
    set mylib.append2yr (keep = date nasdaq WD);
run;
```

```
* DROP function;
** drop nasdaq and WD;

data mylib.s_p (drop = nasdaq WD);
    set mylib.append2yr;
run;
```

Section 4: Work with SAS data

4.1) Creating and Redefining Variables

You can create and redefine variables with **assignment statements** in DATA steps using this basic form.

Variable = expression;

The lift side of the equation is a variable name (either new or old variable), while the right side of the equation can be a constant, another variable, or a mathematical expression. The basic types of assignment statements are:

Assignment Statement	Type of expression
NewVAR = 10;	numeric constant
NewVAR = 'ten';	character constant
NewVAR = OldVAR + 20;	addition
NewVAR = OldVAR - 20;	subtraction
NewVAR = OldVAR * 20;	multiplication
NewVAR = OldVAR / 20;	division
NewVAR = OldVAR ** 20;	exponentiation

Note that SAS performs exponentiation first, then multiplication and division, and followed by addition and subtraction. You can use parentheses for assignment statement so that your programs will be a lot easier to read.

```
data mylib.freqfee;
      set mylib.freq;
      fee_mth2 = sales_mth2 * 0.05;
run;
```

This program shows that a new variable named fee_mth2 is equal to 5% of sale amounts in month#2.

Exercise 1:

Use a data set 'mylib.freqfee to create a new data set 'mylib.freqfee2' and calculate:
(1) commission fee for month 1 (variable 'fee_mth1'), which is 5 % of sale amounts in month# 1
(2) total fee, which is the sum of commission fees of sale amounts in month#1 and #2

4.2) Using IF-THEN Statements

You want an assignment statement to apply to some observations but not all under some conditions. This is called conditional logic, and you do it with IF-THEN statements. The form of these statements is:

IF conditions THEN action;

OR

IF condition-1 AND condition-2 THEN action;

OR

IF condition-1 OR condition-2 THEN action;

Within the conditions, you can use the comparison operator as follows.

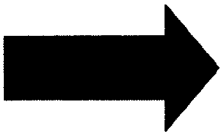
Symbolic	Meaning
=	equals
^=	not equal
>	greater than
<	less than
>=	greater than or equal
<=	less than or equal

An example of if-then statements is:

```
data mylib.freqfee3;
    set mylib.freqfee2;
    if total_fee = . then total_fee = 0;
    if sex ='female' then sexcode = 1;
    if sex ='male' then sexcode = 2;
run;
```

4.3) Grouping Observations with IF-THEN/ELSE Statements

red	xx
orange	xx
green	xx
blue	xx
yellow	xx



red	warm	xx
orange	warm	xx
green	cool	xx
blue	cool	xx
yellow	warm	xx

You can use a series of IF-THEN/ELSE statements to create a grouping variable. The form of these statements is:

IF conditions THEN action;
ELSE IF conditions THEN action;
ELSE IF conditions THEN action;
ELSE action;

Exercise 2:

Use a data set 'mylib.freqfee_ifelse' and write the if-then/else statements to group a 'total_fee' variable by following these conditions:

fee > = 10000 then bonus = 1000
fee > = 5000 then bonus = 500
fee < 5000 then bonus = 0

Note that use mylib.freqfee_ifelse, no need to create new data set!

Brainstorming: How to write the if-then/else statements?

4.4) Using SAS functions with DATA

The following table lists definitions and syntax of commonly used functions.

Function name	Syntax	Definition
INT	INT(argument)	returns the integer portion of argument
LOG	LOG(argument)	natural logarithm
LAG	LAG(argument)	returns the lag
ROUND	ROUND (argument, round-off-unit)	rounds to nearest round-off unit
SUM	SUM (argument, argument,...)	sum of non-missing values
MAX	MAX (argument, argument,...)	largest non-missing value
MIN	MIN (argument, argument,...)	smallest non-missing value

* argument = value, variable name, or expression

4.5) Sorting Data with PROC SORT

The form of this procedure is:

```
DATA  data set-2;
        SET data set-1;
PROC SORT;
        BY variable-1... variable-n;
```

OR

```
PROC SORT DATA = data-set-1 OUT = data-set-2;
        BY variable-1... variable-n;
```

By default SAS sorts data in ascending order, from lowest to highest or from A to Z. To request SAS to sort data from highest to lowest, simply add keyword DESCENDING to the BY statement before each variable that should be sorted.

4.6) Printing Data with PROC PRINT

PROC PRINT prints all variables for all observations in the SAS data set. SAS decides the best way to format the output so you do not have to worry about things like how many variables will fit on a page.

The form of the PRINT procedure is:

```
PROC PRINT DATA = data-set;
        title 'the title of the report';
        footnote 'the footnote of the report';
```

OR

```
PROC PRINT DATA = data-set;
        VAR variable1... variable n;
        title 'the title of the report';
        footnote 'the footnote of the report';
```

4.7) Counting Data with PROC FREQ

PROC FREQ is used to create tables showing the distribution of categorical data values. When you have counts for one variable, they are called one-way frequencies. When you combine two or more variables, the counts are called two-way, three-way, and so on up to n-way frequencies.

The form of the frequency procedure is:

```
PROC FERQ DATA = data-set;
        TABLES variable-combinations;
```

Example:

```
* run proc frequency;

      Proc freq data = mylib.freqfee3;
         tables bonus sex*bonus;
      run;

Note: you can use mylib.freqfee_ifelse instead.
```

Section 5: Using Basic Statistics

5.1) PROC UNIVARIATE

This procedure produces statistics describing the distribution of a single variable. These statistics include the mean, median, mode, standard deviation, and skewness.

The form of this procedure is:

```
PROC UNIVARIATE DATA = data-set;
        VAR variable-1 ... variable n;
```

Example:

```
* run proc univariate;

      Proc univariate data = mylib.univariate;
         var math_sc;
      run;
```

Output from SAS:

The UNIVARIATE Procedure
Variable: MATH_SC (MATH_SC)

Moments

N	17	Sum Weights	17
Mean	69.0882353	Sum Observations	1174.5
Std Deviation	6.98376531	Variance	48.7729779
Skewness	0.35097202	Kurtosis	-0.7795626
Uncorrected SS	81924.5	Corrected SS	780.367647
Coeff Variation	10.1084726	Std Error Mean	1.69381189

Basic Statistical Measures

Location		Variability	
Mean	69.08824	Std Deviation	6.98377
Median	69.50000	Variance	48.77298
Mode	65.00000	Range	22.75000
		Interquartile Range	9.50000

NOTE: The mode displayed is the smallest of 2 modes with a count of 2.

Quantiles (Definition 5)

Quantile	Estimate
100% Max	82.50
99%	82.50
95%	82.50
90%	79.75
75% Q3	74.50
50% Median	69.50
25% Q1	65.00
10%	60.00
5%	59.75
1%	59.75
0% Min	59.75

Output from SAS (continued):

The UNIVARIATE Procedure
Variable: MATH_SC (MATH_SC)

Extreme Observations

-----Lowest----		----Highest----	
Value	Obs	Value	Obs
59.75	1	74.50	12
60.00	8	75.50	5
60.50	4	76.75	15
61.50	3	79.75	16
65.00	6	82.50	9

5.2) PROC MEANS

Most of the descriptive statistics that you can produce with PROC UNIVARIATE you can also generate with PROC MEANS, but you have to ask for them. UNIVARIATE is useful when you know you want all the summary statistics because this procedure prints all these statistics by default. But if you want a few of these statistics then the MEANS procedure is a better way to use.

The MEANS procedure is:

```
PROC MEANS DATA = data-set statistic-keywords;
```

OR

```
PROC MEANS DATA = data-set statistic-keywords;
    VAR variable-list;
```

If you do not include any keywords, then this procedure will produce the mean, the number of non-missing values, the standard deviation, and the minimum and the maximum value for each numeric variable. Some of statistic keywords are:

MAX	Maximum value	STDERR	Standard error of the mean
MIN	Minimum value	P1	1% quantile
N	Number of non-missing values	P5	5% quantile
MEAN	Mean	RANGE	the range
STDDEV	Standard deviation		
VAR	Variance		

Exercise 3:

Use a data set 'mylib.univariate' and perform the MEANS procedure to generate mean, standard deviation, maximum, and minimum for math_sc, eng_sc, and sci_sc variables

5.3) PROC CORR

This procedure is used to compute correlations (i.e., the PEARSON correlations by default). The form of PROC CORR is:

PROC CORR DATA = data-set;

OR

PROC CORR DATA = data-set;
VAR variable-list;
WITH variable-list;

Example:

```
* run proc corr;

 proc corr data = mylib.univariate;
   var math_sc eng_sc sci_sc;
   with math_sc eng_sc sci_sc;
run;
```

Output:

correlations for simple statistics for math_sc, english_sc, and sceince_sc

The CORR Procedure

3 With Variables: MATH_SC eng_sc sci_sc
3 Variables: MATH_SC eng_sc sci_sc

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum	Label
MATH_SC	17	69.08824	6.98377	1175	59.75000	82.50000	MATH_SC
eng_sc	17	69.04412	9.64132	1174	54.00000	85.00000	ENG_SC
sci_sc	17	74.58824	6.98377	1268	65.25000	88.00000	SCI_SC

Pearson Correlation Coefficients, N = 17
Prob > |r| under H0: Rho=0

	MATH_SC	eng_sc	sci_sc
MATH_SC	1.00000	-0.42345	1.00000
MATH_SC		0.0903	<.0001
eng_sc	**-0.42345**	1.00000	-0.42345
ENG_SC	0.0903		0.0903
sci_sc	**1.00000**	-0.42345	1.00000
SCI_SC	<.0001	0.0903	

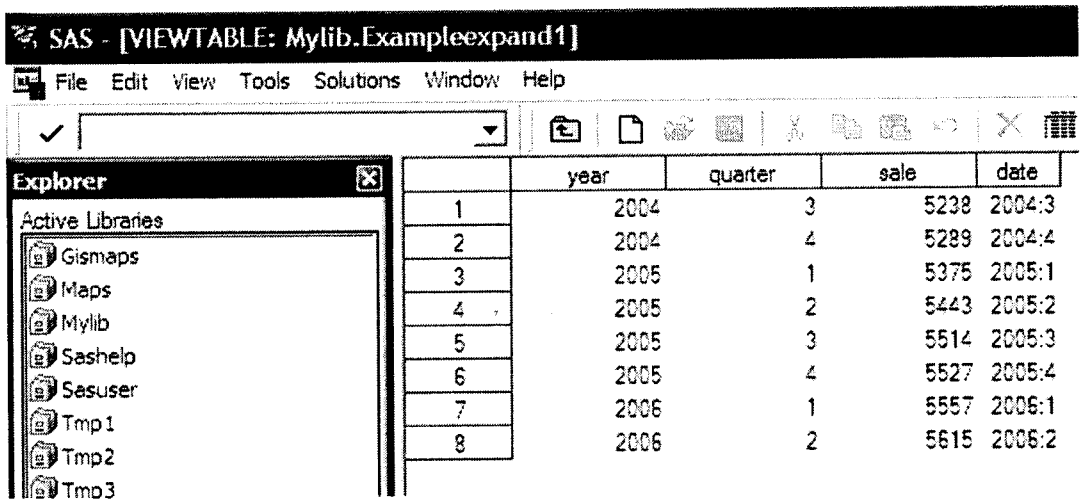
5.4 PROC EXPAND

The EXPAND procedure converts time series from one frequency to another and interpolates missing values in time series. You can collapse time series data from higher frequency intervals to lower frequency intervals, and expand data from lower frequency intervals to higher frequency intervals. For instance, quarterly estimates can be interpolated from an annual series, or quarterly values can be aggregated to produce an annual series.

The expand procedure also supports a wide array of data transformation. Some of the transformation operations are moving average of n neighboring values, backward moving sum, and weighted moving average. Additionally, you can perform LAG or LEAD functions under this procedure.

The following example shows the use of LAG and LEAD functions and the calculation of moving average of the series of quarterly sales (time t, ..., time t-n).

The raw data of the series of quarterly sales during 2004-2006 is shown below.



	year	quarter	sale	date
1	2004	3	5238	2004:3
2	2004	4	5289	2004:4
3	2005	1	5375	2005:1
4	2005	2	5443	2005:2
5	2005	3	5514	2005:3
6	2005	4	5527	2005:4
7	2006	1	5557	2006:1
8	2006	2	5615	2006:2

PROGRAM:

```
proc expand data= mylib.exampleexpand1 out=mylib.outexpand method = none;
     id date;
        convert sale =slag_2 / transform=(lag 2);
        convert sale =slag_1 / transform=(lag 1);
        convert sale;
     convert sale =slead_1 / transform=(lead 1);
     convert sale =slead_2 / transform=(lead 2);
     convert sale =s_movavg / transform=(movave 3);
   run;
```

OUTPUT:

Help

Sales Moving Average 13:28 Thursday, April 19, 2007

Obs	date	slag_2	slag_1	sale	slead_1
1	2004:3	.	.	5238	5289
2	2004:4	.	5238	5289	5375
3	2005:1	5238	5289	5375	5443
4	2005:2	5289	5375	5443	5514
5	2005:3	5375	5443	5514	5527
6	2005:4	5443	5514	5527	5557
7	2006:1	5514	5527	5557	5615
8	2006:2	5527	5557	5615	.

Obs	slead_2	s_movavg	year	quarter
1	5375	5238	2004	3
2	5443	5263.5	2004	4
3	5514	5300.6666667	2005	1
4	5527	5369	2005	2
5	5557	5444	2005	3
6	5615	5494.6666667	2005	4
7	.	5532.6666667	2006	1
8	.	5566.3333333	2006	2

Integrated Exercise:

Using a data set named 'mylib.integratedtest', you are requested to assign a final grade (i.e., A , B+, B, C+, and C) to individual students. The scores of individual students for a midterm exam, a final exam, and a project are shown below.

idno	name	sex	midterm	final	project
4602610011	papada	female	15.25	27	17.5
4602610022	sasi	female	16.5	30	18.5
4602610024	nana	female	17	25	19.5
4602610029	sutee	male	19	26	15.5
4602610121	tanapol	male	25	34	16.5
4602610333	penporn	female	17	30.5	17.5
4602610342	teerachart	male	24.5	29.5	16.5
4602611223	santi	male	18.5	25	16.5
4602611567	chomdoa	female	26	36	20.5
4602680033	weena	female	18	28	20.5
4602680045	nuntakarn	female	23.5	32.75	14.5
4602680123	nutchai	male	22	31	21.5
4602682890	pattima	female	21	27	22.5
4602683500	napat	male	22	24.5	23
4602683623	voravut	male	25	35	16.75
4602684101	nuttra	female	24	36.5	19.25
4602684203	jariya	female	21.5	28	16.5

For writing a SAS program for grading, here are the steps that you may follow:

1. Calculate new variable named **total** (total = the sum of midterm, final, and project)
2. Round up values of the total variable, and Name the new variable as **roundtotal**

3. Use PROC MEANS for calculating mean, maximum, minimum, and standard deviation of a roundtotal variable
4. Perform PROC FREO to check the frequency of a roundtotal variable
6. Assume that:

   Student gets A if roundtotal > = 80
   B+ if roundtotal > = 75
   B if roundtotal > = 70
   C+ if roundtotal > = 65 and
   C if roundtotal < 65

   My suggestion is that use the IF-THEN/ ELSE statements. Do not forget to sort data (BY a roundtotal) before running IF-THEN/ ELSE statements.

7. Perform PROC FREQ to check the frequency of a roundtotal variable, so you can fill in the number of students who receive A, B+, B, C+ and C in the grade form
8. Print the grade report

   My suggestion is that use the VAR variable-list statement in PROC PRINT if you want to print only idno and grade

...The End...

References:

Delwiche, L.D. and S. J. Slaughter. 1996. *The Little SAS Book: A Primer.* Cary, NC. SAS Institute Inc.

Online SAS documents

การอบรม

เรื่อง

"Introduction to SAS Programming"

โดย

รองศาสตราจารย์ ดร.พันทิศา ภาวบุตร

วันพฤหัสบดีที่ 26 เมษายน 2550
เวลา 09:00 – 10:30 น.
เวลา 13:00 – 16:00 น.
ศูนย์คอมพิวเตอร์ ชั้น 2

ศูนย์วิจัยธุรกิจ
คณะพาณิชยศาสตร์และการบัญชี
มหาวิทยาลัยธรรมศาสตร์

Introduction to SAS Programming

THAMMASAT
BUSINESS SCHOOL

Speakers: Piyaratt Jantadej and Pantisa Pavabutr

Program Assistant: Anutchanat Jaroenjitrkam

April 26, 2007

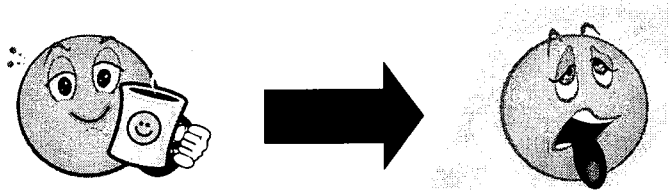
Class Plan: April 26th, 2007

Morning Session:

- Binary models
- Ordinary least squares

Afternoon session:

- Principal components analysis
- Writing macros

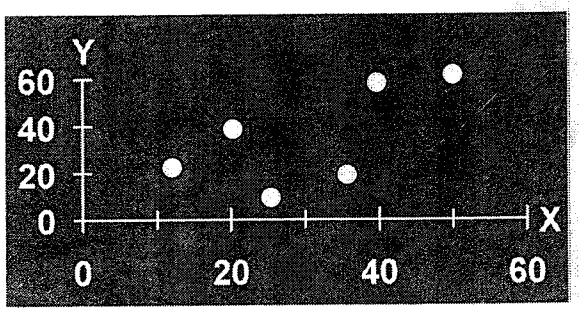


Regression Modeling Steps

1. Hypothesize Deterministic Component
2. Estimate Unknown Model Parameters
3. Specify Probability Distribution of Random Error Term
   Estimate Standard Deviation of Error
4. Evaluate Model
5. Use Model for Prediction & Estimation

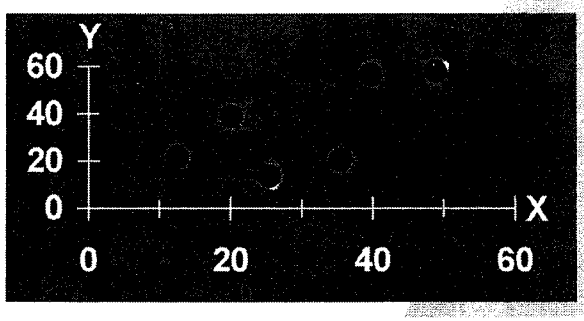
Scattergram

- 1.Plot of All (X_i, Y_i) Pairs
- 2.Suggests How Well Model Will Fit



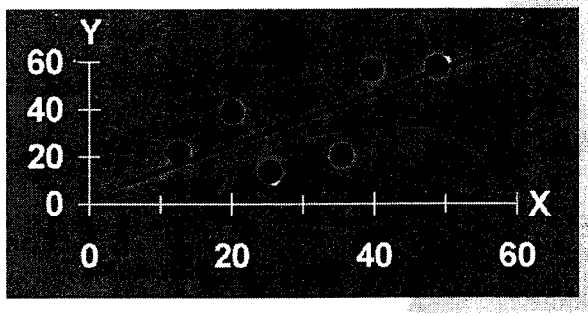
Thinking Challenge

How would you draw a line through the points? How do you determine which line 'fits best'?



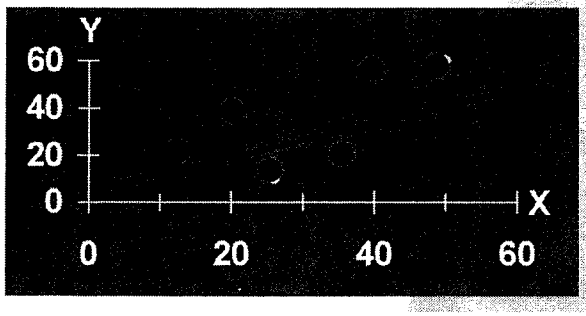
Thinking Challenge

How would you draw a line through the points? How do you determine which line 'fits best'?



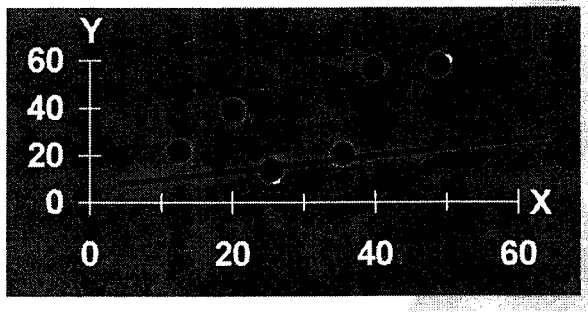
Thinking Challenge

How would you draw a line through the points? How do you determine which line 'fits best'?



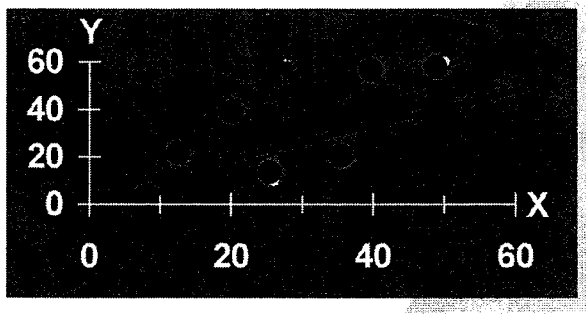
Thinking Challenge

How would you draw a line through the points? How do you determine which line 'fits best'?



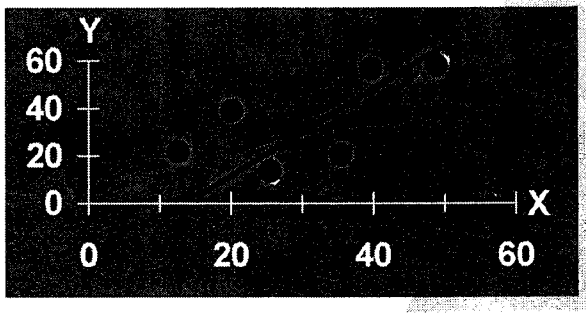
Thinking Challenge

How would you draw a line through the points? How do you determine which line 'fits best'?



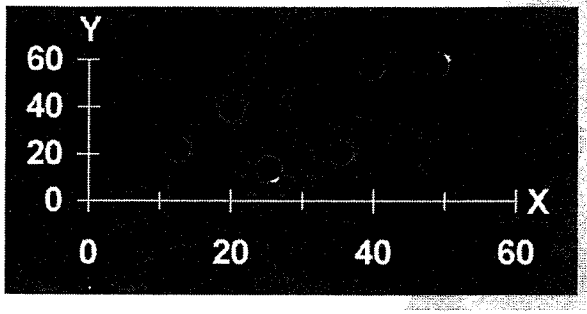
Thinking Challenge

How would you draw a line through the points? How do you determine which line 'fits best'?



Thinking Challenge

How would you draw a line through the points? How do you determine which line 'fits best'?



Least Squares

- 1.'Best Fit' Means Difference Between Actual Y Values & Predicted Y Values Are a Minimum
  - – *But* Positive Differences Off-Set Negative

Least Squares

- 1.'Best Fit' Means Difference Between Actual Y Values & Predicted Y Values Are a Minimum
  - *But* Positive Differences Off-Set Negative

$$\sum_{i=1}^{n}\left(Y_i-\hat{Y}_i\right)^2=\sum_{i=1}^{n}\hat{\varepsilon}_i^2$$

Least Squares

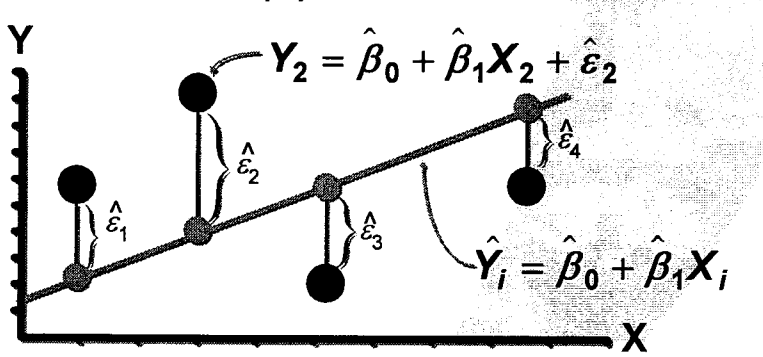
- 1.'Best Fit' Means Difference Between Actual Y Values & Predicted Y Values Are a Minimum
  - *But* Positive Differences Off-Set Negative

$$\sum_{i=1}^{n}\left(Y_i-\hat{Y}_i\right)^2=\sum_{i=1}^{n}\hat{\varepsilon}_i^2$$

- 2.LS Minimizes the Sum of the Squared Differences (SSE)

Least Squares Graphically

LS minimizes $\sum_{i=1}^{n}\hat{\varepsilon}_i^2=\hat{\varepsilon}_1^2+\hat{\varepsilon}_2^2+\hat{\varepsilon}_3^2+\hat{\varepsilon}_4^2$



Coefficient Equations

Prediction Equation

$$\hat{y}_i=\hat{\beta}_0+\hat{\beta}_1x_i$$

Sample Slope

$$\hat{\beta}_1=\frac{SS_{xy}}{SS_{xx}}=\frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sum(x_i-\bar{x})^2}$$

Sample Y-intercept

$$\hat{\beta}_0=\bar{y}-\hat{\beta}_1\bar{x}$$

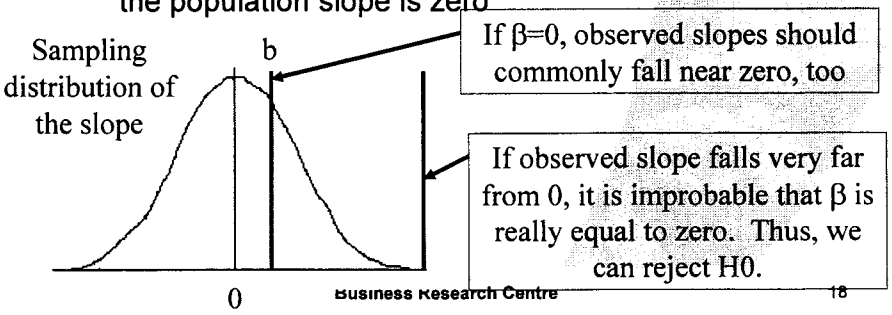
Interpretation of Coefficients

Slope ($\hat{\beta}_1$)

- Estimated *Y* Changes by $\hat{\beta}_1$ for Each 1 Unit Increase in *X*
  - If $\hat{\beta}_1$ = 2, then dependent variable (*Y*) Is Expected to Increase by 2 for Each 1 Unit increase in explanatory variable (*X*)

Review: Slope Hypothesis Tests

- Visually: If the population slope (β) is zero, then the sampling distribution would center at zero
  - Since the sampling distribution is a **probability distribution**, we can identify the likely values of **b** if the population slope is zero



Regression Hypothesis Tests

- If assumptions are met, the sampling distribution of the slope (b) approximates a T-distribution
- Standard deviation of the sampling distribution is called the standard error of the slope (σ_b)
- Population formula of standard error:

$$\sigma_b = \sqrt{\frac{\sigma_e^2}{\sum_{i=1}^{N}(X_i - \bar{X})^2}}$$

- Where σ_e^2 is the variance of the regression error

Regression Hypothesis Tests

- Estimating σ_e^2 lets us estimate the standard error:

$$\hat{\sigma}_e = \sqrt{\frac{\sum_{i=1}^{N} e_i^2}{N-2}} = \sqrt{\frac{SS_{ERROR}}{N-2}} = \sqrt{MS_{ERROR}}$$

- Now we can estimate the S.E. of the slope:

$$\hat{\sigma}_b = \sqrt{\frac{MS_{ERROR}}{\sum_{i=1}^{N}(X_i - \bar{X})^2}}$$

Regression Hypothesis Tests

- Finally: A t-value can be calculated:
  - It is the slope divided by the standard error

$$t_{N-2} = \frac{b_{YX}}{s_b}$$

- Where s_b is the sample point estimate of the standard error
- The t-value is based on N-2 degrees of freedom

Proc REG Syntax

PROC REG <options> ;
MODEL dependent = <regressors> </options> ;
BY variables ;
OUTPUT <OUT = datasets> ;
RUN ; QUIT ;

A simple OLS regression in SAS

```
proc reg data=gold outest=coeff;
model nyse = cpi ;
run ; quit ;

proc reg data=gold noprint ;
model nyse = cpi  /noint ;
run ; quit ;

proc reg data=gold ;
model goldprice nyse = CPI ;
output out=parem
p = goldpricehat nysehat
r = goldpriceresid nyseresid ;
run; quit ;
```


How do we obtain "BLUE"

- Zero mean value of disturbance term
- Homoscedasticity of disturbance term
- No autocorrelation in disturbances
- No perfect multicollinearity
- Model correctly specified
- Model is linear in parameter

Zero mean value of disturbance term

- Detecting: $E(u_i \mid X_i) = 0$

proc reg data=gold noprint outest=coeff;
model nyse = cpi ;
plot r.*cpi ;
run ; **quit** ;

- Residuals should have no apparent trend otherwise you should revise your model.

Homoscedasticity of disturbance

- Detecting $\text{var}(u_i \mid X_i) = \sigma^2$

proc reg data=gold noprint outest=coeff;
model nyse = cpi ;
plot r.*p. ;
run ; **quit** ;

- The residual should show no discernable pattern.
- Otherwise, use more formal chi-square test : Null: Homoscedasticity

proc reg data=gold outest=coeff;
model nyse = cpi /spec ;

run ; **quit** ;

No autocorrelation in disturbances

- Detecting: Use Durbin Watson Stats $\hat{\rho} = \dfrac{\sum \hat{u}_t \hat{u}_{t-1}}{\sum \hat{u}_t^2}$

$$d = 2(1 - \hat{\rho})$$

proc reg data=gold outest=coeff;
model nyse = cpi /dw ;

run ; **quit** ;

- As a rule of thumb, if d is approx 2.0 then you may assume no first order autocorrelation, but the closer d is to 0 (4), the greater the evidence, the greater the chances of positive (negative) FO autocorrelation. More formally, look up DW table.

No multicollinearity

- This usually leads to high Rsq but low t-stats on parameter estimates
- Detecting with tolerance and variance inflation factor

$$TOL_j = (1 - R_j^2) = 1/(VIF_j)$$

proc reg data=gold outest=coeff;
model goldprice = cpi nyse / tol vif ;
run ; **quit**;

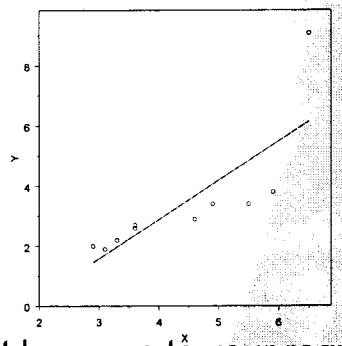
- The Rsq computed in TOL is from auxiliary regression of regressor Xj on the remaining (k-2) regressors. Clearly, if TOTLj is close to 0 or VIF exceeds 10.0, the variable is likely to be highly collinear.

Fixing the problems

Problem	Possible solution
Non-zero mean disturbance	Check if it is HSK or autocorrelation problem
HSK	Use GLS or White correction (Refer to proc model)
Autocorrelation	Remodel the error specification (Refer to proc autoreg)
Multicollinearity	Re-specify model or use instruments

Getting rid of outliers

Y	X
3.4	5.5
3.8	5.9
9.1	**6.5**
2.2	3.3
2.6	3.6
2.9	4.6
2.0	2.9
2.7	3.6
1.9	3.1
3.4	4.9

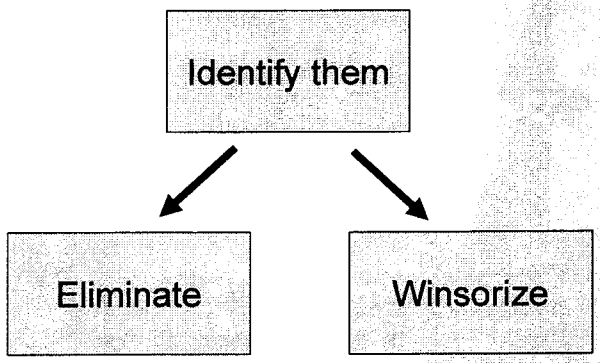


What happens to your regression estimates with and without outliers?

Try this

```
data outlier ;
input Y X ;
datalines;
3.4  5.5
3.8  5.9
9.1  6.5
2.2  3.3
2.6  3.6
2.9  4.6
2.0  2.9
2.7  3.6
1.9  3.1
3.4  4.9
;

proc reg data=outlier ;
model y = x ; run; quit ;
```


How to get rid of outliers



Identifying outliers in SAS

proc sort data=nasm ; by date ; **run ;**

proc univariate data=nasm round = **0.001** noprint ;
var ret ; by date ;
output out= extr pctlpts= **5 95** pctlpre=ret ; **run ;**

Elimination

data elim ; merge extr nasm; by date ; **run ;**

proc sort data=elim ; by ccode date ; **run ;**

data elim; set elim ;
if ret le ret5 then ret = . ; else if ret ge ret95 then ret = . ; **run ;**

Winsorization

data win ; merge extr nasm; by date ; **run ;**

proc sort data=win ; by ccode date ; **run ;**

data win ; set win ;
if ret le ret5 then ret = ret5 ; else if ret ge ret95 then ret = ret95 ; **run ;**

Factor analysis

- **Factor analysis** is a statistical data reduction technique used to explain variability among observed random variables in terms of fewer unobserved random variables called **factors**. The observed variables are modeled as linear combinations of the factors, plus "error" terms.
- Factor analysis originated in psychometrics, and is used in behavioral sciences, social sciences, marketing, product management, operations research, and other applied sciences that deal with large quantities of data.

Advantages/Disadvantages

Pros

- Reduced dimensionality
- Takes care of latent variables

Cons

- Lack of economic or theoretical grounds. Interpreting factor analysis is based on using a "heuristic", which is a solution that is "convenient even if not absolutely true"
- Factor analysis can not identify causality.

Approaches to factor analysis

- There are two approaches to factor analysis: "principal component analysis" (the total variance in the data is considered); and "common factor analysis" (the common variance is considered).
- In the common factor model, unique variables are required to be uncorrelated, whereas residuals in principal components are correlated.
- In PCA, the number of components will be original to the number of the original variables.

Factor analysis with PCA

- Let $r = \begin{bmatrix} r_1 \\ r_2 \\ ... \\ r_n \end{bmatrix}$ be n asset returns.
- Let $\sum$ be the covariance matrix of r, so that $\sum$ is an n x n matrix.
- We believe that there are n independent (unobservable) factors, $F_1, F_2, ..., F_n$, which determines r.
- Hence r = Bf or $F = A \cdot r$

Factor analysis with PCA

- Then the covariance of F is, $\Sigma_f = A' \Sigma A$
- Since we want the factors to be orthogonal, we need covariance of all factors to be zero and the matrix diagonal.
- Eigenvalues and eigenvectors serve this purpose.

A less abstract case..Student Entrance Score

Two tests VERBAL and QUANT have a correlation of 0.54662.

The principal component solution is

Eigenvalues (Factor loadings)		
	1	2
Verbal	0.87938	0.47612
Quant	0.87938	-0.47612

Eigenvector (Factor score)		
Verbal	0.569	1.05
Quant	0.569	-1.05

Factors	1	2
Var explained by components	1.55	0.45
% of total variance explained	77.33	22.67

Interpretation

- VERBAL = 0.87938PC1 + 0.47612PC2
- QUANT = 0.87938PC1 – 0.47612PC2
- OR
- PC1 = 0.569*VERBAL + 0.569*QUANT
- PC2 = -1.05*VERBAL+1.05*QUANT

A number of relationships can be observed,

- Var(VERBAL) = Var(0.87938*PC1) + Var(0.47612*PC2)
- = 0.87938^2Var(PC1) + 0.47612^2Var(PC2)
- = 1
- Similarly, VAR(QUANT) = 1

Interpretation

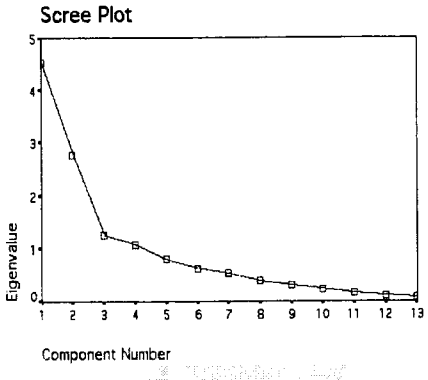
- The variance explained by the first component is.
- $0.87938^2 + 0.87938^2 = 1.54662$ (Eigenvalue of PC1)
- The variance explained by the second component is,
- $0.47612^2 + 0.47612^2 = 0.45338$ (Eigenvalue of PC2)

- So the variance explained by sum of the first two components is 1.54662 + 0.45338 = 2.0

- Thus, percentage variance explained by PC1 = 1.54662/2 = 77.33%
- Percentage variance explained by PC2 = 0.4553/2 = 22.67%

How many factors should be retained?

- One rule is to consider only those with eigenvalues over 1.
- Another rule of thumb is to plot all the eigenvalues in their decreasing order. The plot looks like the side of a mountain, and "scree" refers to the debris fallen from a mountain and lying at its base.



Proc factor: Syntax

Proc factor data= datafile <options> ;
Var <variables> ;
By <variables> ;

proc factor scree data=score method=principal
nfactors=**3** outstat= outstat;
var exam1 exam2 exam3;
run ;

Proc factor

proc factor scree data=score method=principal
nfactors=**3** outstat= outstat;
var exam1 exam2 exam3;
run ;

proc factor scree data=score method=ml
rotate=varimax nfactors=**3** outstat= outstat;
var exam1 exam2 exam3;
run ;

Writing macros

- What are macros?
- A SAS macro is way of defining parts of or collections of SAS statements which can be carried out repeatedly or which can substitute names of datasets and variables for symbolic names.

Program with macro and without macro

Without

```
proc append base=all data=file_1 ; run;
proc append base=all data=file_2 ; run;
proc append base=all data=file_2 ; run;

.....

proc append base=all data=file_n-1 ; run;
proc append base=all data=file_n ; run;
```

With

```
%macro appfile ;  %do i=1 %to n ;
proc append base=all data=file&i ;

%end ; %mend appfile ; %appfile ; run
    ;quit ;
```

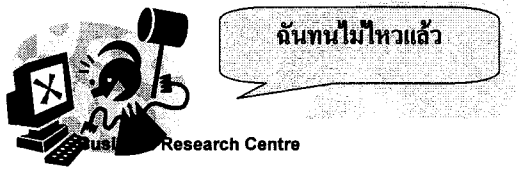
Program with and without macro

Without

```
proc means data=Korea1 ;
var ret
title "Univariate Procedure: country=Korea
    Period =1 ;
run ;

proc means data=Korea2 ;
var ret
title "Univariate Procedure: country=Korea
    Period =2 ;
run ;
```

With

```
%macro first;
  %let allc= korea thai taiw ;
    %do i = 1 %to 3;
%let cvar = %scan(&allc,&i,' ');
     %do j = 1 %to 5 ;
proc means data=&cvar&i ;
      var ret ;
title "Univariate Procedure: country=&cvar
    period=&j";
 run;
%end; %mend first; %first ; run ; quit;
```



General macro syntax

Starts with:

%macro <macro name> ; **%do** i=1 %to **n** by **m ; %put &i**;

%macro reg(data=, y=, x=);

Ends with:

%end ; **%mend** <macro name> ; %macroname **; run ; quit ;**

Recalling macros

%reg(data=mydata, y=salary, x=educ);

%reg(data=mydata, y=salary, x=age);

%reg(data=mydata, y=salary, x=sex);

Example

- **%macro** dataread(year);
- data temp; set bngkset; if year=&year;
- proc means data=temp; var ret volb ;
- title "Mean of &year" ;
- run;
- **%mend** dataread;

- %***dataread***(**1995**) ;
- %***dataread***(**1997**) ;
- %***dataread***(**1999**) ;

Example

- **%macro** reg(data= , y=, x=);
- proc reg data=&data;
- model &y = &x;
- title "Reg of &y on &x" ;
- plot p. * r.; **%mend**;

- %***reg***(data=bngkset, y=ret, x=volb);
- %***reg***(data=bngkset, y=ret, x=volsh); **run** ; **quit** ;

Q&A

เฮ้อ.....จบเสียที

Fun SAS Activities

Morning Activities (Apr 26, 2007)

Activity 1

1. Import file SP500.csv into SAS
2. Run regression of SP500 on NASDAQ and check for autocorrelation
3. Rerun the regression in 2 without intercept and check for autocorrelation

Activity 2

1. Import file Indus1.csv into SAS
2. Keep variable yrmo mkt smb hml and drop all other variables
3. Rename mkt variable as Y, smb as X1 and hml as X2
4. Compute "Pearson" correlation matrix of Y, X1 and X2.
5. Run regression of Y on X1 and X2 and check for HSK and multicollinearity
6. Create coefficient output table as sas dataset and call it coeff1

Activity 3

1. Run regression of Y on X1 and create output table as sas dataset and call it coeff2
2. Append dataset coeff1 and coeff2

Afternoon Activities (Apr 26, 2007)

Files used

SP500.csv
Indus1.csv

Activity 1
Use the file SP500 to determine the degree of commonality between NASDAQ and SP500. Do they have a common factor?

For activities 2, 3, and 4 please write an appropriate macro where necessary.

Activity 2
Use the Indus1.csv file to run regressions of variables (please write a macro for this)

a) mkt on smb
b) mkt on hml
c) mkt on smb and hml

Activity 3

1. Read the first 60 observations from file indus1.csv into a new sas datafile (Hint: Use command <id=_n_>)

2. Run a "rolling regression" of mkt on smb and hml using the 1st to 20th observation, then the 2nd to 21st observation, then 3rd to 22nd observation and so on.

3. Output and append the coefficients of each regression in one sas dataset file.

A Synthesis...
For the following activities these files are used.

Files used:
1. SAS exercise1.xls (DJAIG, MSCIACworld, SPHG)
   - DJAIG: Dow Jones Commodity Index
   - MSCI AC world Index: MSCI All Country Index
   - SPHG: S&P Hedge Fund Index
2. SAS exercise2.xls (SP500, Nasdaq)
   - S&P 500 Index
   - Nasdaq Index

Instructions:
1. Import data files to SAS program.
2. Merge these two files by date.
3. Change date format from MM/DD/YYYY to DD/MM/YY.
4. Rename MSCIACworld to MSCI.
5. Calculate return of indices.
6. Find outliners and eliminate (if any).
7. Plot graph of ret_SP500 (Y) and ret_SPHG (X).
8. Run regression equation of :
   a. ret_MSCI (t) = a0 + a1*ret_MSCI (t-1) + a2*ret_MSCI (t-2) + e(t)
   b. ret_SP500 = a0 + a1*ret_SPHG + a2*ret_DJAIG
9. Do as (7) but select only period of 2005.

การอบรม

เรื่อง

"Introduction to SAS Programming"

โดย

อาจารย์ ดร.ปิยรัตน์ จันทะเดช

วันพฤหัสบดีที่ 26 เมษายน 2550
เวลา 10:30 – 12:00 น.
ศูนย์คอมพิวเตอร์ ชั้น 2

ศูนย์วิจัยธุรกิจ
คณะพาณิชยศาสตร์และการบัญชี
มหาวิทยาลัยธรรมศาสตร์

THAMMASAT
BUSINESS SCHOOL

Agenda

- What is Binary Logistic Regression Model?
- Problems of OLS
- Logistic function
- Assumptions of Logistic Model
  - Diagnose Multicollinearity
- SAS program
- Exercise and Output Interpretation

Binary Logistic Regression Model

- Binary regression is a form of regression that is used when **the dependent variable is a dichotomy** and the independent variables are of any type (continuous or categorical)
- The dependent variable is set up as a 0-1 dummy variable
  - Success/ Failure
  - Absence/Presence
  - Buying/Not Buying

Problems of OLS

- Linear Model:

$$E(Y/X_1,\ldots.X_k) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots. + \beta_k X_k$$

Note that **the RHS** of model can take any value between $-\infty$ and $+\infty$

Problems of OLS (2)

- After regressed a 0-1 dependent variable (DV) on the independent variables (IDVs), we would **expect the predicted values of the dependent variable** to fall mainly within the interval between 0 and 1
  - Predicted value of DV could be **interpreted as the probability** such as the probability of the presence of disease, given that IDVs' characteristics (age, genetic code)

Problems of OLS (3)

- HOWEVER when Y takes value of 0 or 1, it is common for predicted values generated ***by linear model*** to be **outside** the (0, 1) interval because RHS takes value between - ∞ and + ∞

Problems of OLS (4)

- Linear Model
  - $Y = a + bX + e$
  - $E(e) = 0$
  - $VAR(e) = \sigma^2$
  - $COV(e_i, e_j) = 0$
  - $e_i \sim$ Normal
- When DV takes value of 0 or 1, some assumptions (i.e., $VAR(e) = \sigma^2$ and $e_i \sim$ Normal) are violated

Problems of OLS (5)

- Consequences:
  - The coefficient estimates are *no longer efficient* (other estimation methods provide the smaller standard errors)
  - The standard error are *no longer consistent estimates* of the true standard errors

Logistic Function

- What we need is some means of squeezing the estimated probabilities inside the 0-1 interval
  - The logistic function provides the probability that the event will occur (such as the presence of disease) and one minus this function provides the probability that the event will not occur (such as the absence of disease)
  - The function predicts the probability that the event is equal to 1 [ie., Pr $(Y=1/X_1,....X_k)$] in term of **the log odds ratio**

Logistic Function (2)

$$\log\left[\frac{P_i}{1-P_i}\right] = \alpha + \beta_1 X_1 + \beta_2 X_2 + ...+ \beta_k X_k \quad ---(1)$$

where:

$\frac{P_i}{1-P_i}$ = odds = $\frac{\text{probability of event (Y=1)}}{\text{probability of no event (Y=0)}}$

$\log\left[\frac{P_i}{1-P_i}\right]$ = the log odds = the natural logarithm of the odds

$$\log [pr\ (y=1)/pr\ (y=0)] = \alpha + \beta_1 X_1 + \beta_2 X_2 + ...+ \beta_k X_k \quad ---(2)$$

$$Pr\ (y=1) = \frac{\exp (\alpha + \beta_1 X_1 + \beta_2 X_2 + ...+ \beta_k X_k)}{1 + \exp (\alpha + \beta_1 X_1 + \beta_2 X_2 + ...+ \beta_k X_k)} \quad ---(3)$$

$$Pr\ (y=1) = \frac{1}{[1 / (1 + \exp \{- (\alpha + \beta_1 X_1 + \beta_2 X_2 + ...+ \beta_k X_k)\}]} \quad ---(4)$$

Logistic Function (3)

- Wald Statistics are used to test for the significance of individual coefficients (b_i)
- We can interpret the marginal effect of individual coefficients (b_i) on the probability of the occurrence of event [i.e., Pr (Y=1)]
  - In term of exp (b)
  - In term of percentage change:

    That is, [100 * (1- exp (b)]

Assumptions of Logistic Model

- Normally distributed error terms are <u>not</u> assumed
- Homogeneity of variance is <u>not</u> assumed
- Linearity
  - A linear relationship between the DV and the IDVs is <u>not</u> assumed
  - A linear relationship between the IDVs and the log odds of the DV is assumed
- No multicollinearity
  - Use Tolerance and Variance Inflation Factor to detect
- No outliers

SAS program

```
PROC LOGISTIC DATA = data-set DESCENDING;
    MODEL DV = IDV-list;
RUN;
```


Logistic Regression Model Exercise

Scenario

Suppose that we are working with some doctors on heart attack patients. The dependent variable is whether the patient has had a second heart attack within one year (yes= 1; no= 0). We have two independent variables. The first variable is whether the patient completed a treatment consistent of anger control practices (yes=1; no= 0). The other independent variable is a score on a trait anxiety scale (a higher score means more anxious).

A data set is mylib.examplelogit.

Person	2nd Heart Attack	Treatment of Anger	Trait Anxiety
1	1	1	70
2	1	1	80
3	1	1	50
4	1	0	60
5	1	0	40
6	1	0	65
7	1	0	75
8	1	0	80
9	1	0	70
10	1	0	60
11	0	1	65
12	0	1	50
13	0	1	45
14	0	1	35
15	0	1	40
16	0	1	50
17	0	0	55
18	0	0	45
19	0	0	50
20	0	0	60

(Source: http://luna.cas.usf.edu/~mbrannic/files/regression/Logistic.html)

PROGRAM: Multicollinearity Diagnostics from PROC REG

```
proc reg data=mylib.examplelogit;
   model attack = anger anxiety / TOL VIF;
   title 'diagnose multicollinearity';
run;
```

OUTPUT: Multicollinearity Diagnostics from PROC REG

diagnose multicollinearity

The REG Procedure
Model: MODEL1
Dependent Variable: Attack

Number of Observations Read	20
Number of Observations Used	20

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	1.89623	0.94811	5.19	0.0174
Error	17	3.10377	0.18257		
Corrected Total	19	5.00000			

Root MSE	0.42729	R-Square	0.3792
Dependent Mean	0.50000	Adj R-Sq	0.3062
Coeff Var	85.45757		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	**Tolerance**	**Variance Inflation**
Intercept	1	-0.62950	0.46854	-1.34	0.1968	.	0
Anger	1	-0.17410	0.19746	-0.88	0.3902	**0.94601**	**1.05708**
Anxiety	1	0.02110	0.00751	2.81	0.0120	**0.94601**	**1.05708**

PROGRAM for Binary Logistic Regression Model:

```
* run logistic model for mylib.examplelogit;

proc logistic data= mylib.examplelogit descending;
   model attack = anger anxiety;
run;
```

OUTPUT for Logistic Model:

The LOGISTIC Procedure

Response Variable	Attack
Number of Response Levels	2
Model	binary logit
Optimization Technique	Fisher's scoring

Number of Observations Read	20
Number of Observations Used	20

Response Profile

Ordered Value	Attack	Total Frequency
1	1	10
2	0	10

Probability modeled is Attack='1'.

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	29.726	24.820
SC	30.722	27.808
-2 Log L	27.726	18.820

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	8.9055	2	0.0116
Score	7.5849	2	0.0225
Wald	5.3847	2	0.0677

The LOGISTIC Procedure

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-6.3634	3.2139	3.9203	0.0477
Anger	1	-1.0241	1.1711	0.7647	**0.3818**
Anxiety	1	0.1190	0.0550	4.6884	**0.0304**

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
Anger	**0.359**	0.036	3.565
Anxiety	**1.126**	1.011	1.255

Association of Predicted Probabilities and Observed Responses

Percent Concordant	84.0	Somers' D	0.720
Percent Discordant	12.0	Gamma	0.750
Percent Tied	4.0	Tau-a	0.379
Pairs	100	c	0.860

Interpretation:

variable	parameter estimate (B)	odd ratios [exp(B)]	percentage change of odd ratios [exp(B)-1] *100
Anger (dummy variable)	-1.0241	0.359119528	**-64.08804724**
Anxiety (continuous variable)	0.119	1.126369918	**12.63699183**

-64.08 means the odds of experiencing the second heart attack within one year is 64.08 percent lower for person who has treatment of anger than for person who does not have treatment of anger

12.63 means an increase of a score on anxiety scale increases the odds of experiencing the second heart attack within one year by 12.63%, controlling for other variable in the model