STA 4702 Multivariate Statistical Methods

STA 5701 Applied Multivariate Methods

Introduction to SAS

What is SAS?

SAS® is a statistical analysis package. A collection of computer "procedures" and a very powerful data management tool that facilitates data and statistical analysis.

SAS® is not a menu-driven or command driven application. Rather it relies on user-written scripts or "programs" that are processed when requested to know what to do.

Because it is a script-based application, the key to being successful using SAS is learning the rule and tricks of writing scripts.

Rules of SAS Use

0. Every SAS statement ends in a semicolon (;)!!!!!!!!
1. All variable and data set names must start with a letter or an underscore (_).
3. Names can contain only letters, numerals, and the underscore. No %$!*&#@.
4. SAS commands are case insensitive (e.g Body, body, boDY and BoDy are all the same name to SAS).
5. There are no restrictions on where in the line statements start or stop and commands may be wrapped onto multiple lines.
6. All variable and data set names must be thirty-two (32) or fewer characters in length (8 characters in SAS V6.12 or lower).

For this class in order to make SAS programs easier to read, we will (require) attempt to have each new SAS statement start on a new line.

General Form of a SAS Program

SAS programs (scripts) are composed of DATA blocks and PROCedure Blocks.

DATA blocks consists of SAS statements telling the application what to do with the data set (e.g input formats, variable names, create new variables, etc.)

PROC blocks consists of SAS statements telling the applications what to do to the data set (e.g print it out, create a cross-tabulation table, perform computations for a statistical analysis.

```
DATA oranges;
  INPUT state $ 1-10 early 12-14 late 16-18;
  DATALINES;
Florida    124  90
California  40  27
Texas      1.3 .3
Arizona    .4 .5
;
```

DATA STEP

```
PROC PRINT DATA=oranges;
  RUN;
```

PROC STEP

Running SAS

SAS has three run modes.

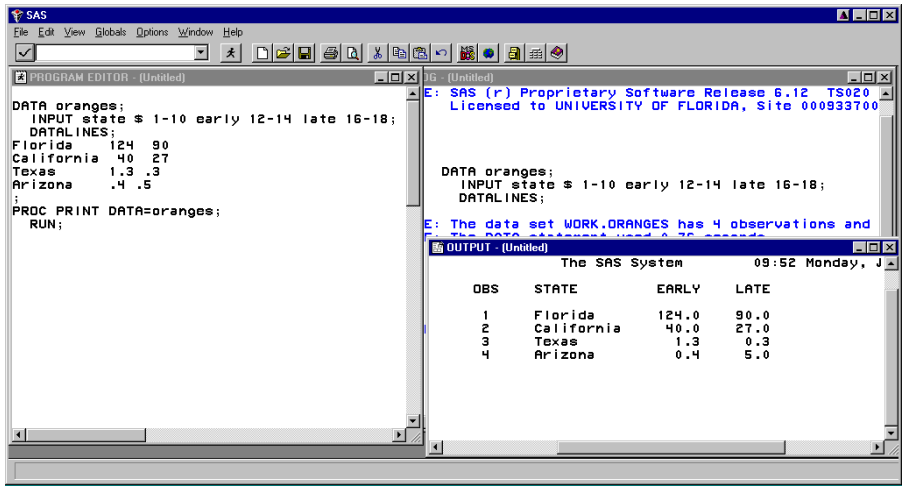
1. Display Manager Mode
2. Interactive Line Mode
3. Batch or background mode.

For this class we will be illustrating examples in **Display Manager Mode**.

Running SAS jobs at CIRCA or NERDC may require learning to use the **Batch mode**. Batch mode requires a multi-processing operating system (UNIX, Linix or OS2) to be effective.

There is NO DIFFERENCE in SAS code between what is run in Display Manager Mode and what is run in Batch Mode. All that is different is how the user interacts with the application.

Display Manager Mode



Different ways one can read in data using SAS

Sections 3.1, 3.2, 3.6, 3.7, 3.8, 3.10, 3.12, 3.13, 3.14
The Little SAS Book.

```
data oranges;
input state $ 1-10 early 12-14 Late 16-18;
datalines;  /* or cards; */
Florida    130  90
California  37  26
Texas      1.3 .15
Arizona    .65 .85
;
```

COLUMN INPUT

Data is included in the SAS script (program).

Reading Data from an External Text File

a:\yields\oranges.dat

```
Projected Orange Yields in October 1997
State    Early Late
Florida    130  90          <- Line 3
California  37  26
Texas      1.3 .15
Arizona    .65 .85          <- Line 6
Based on information obtained from the
Florida Agricultural Statistics Service
```

```
data oranges;
infile 'a:\yields\oranges.dat' firstobs=3 obs=6;
input state $ 1-10 early 12-14 late 16-18;
```

Read 6 lines but only take data starting on line 3.

Reading Data from the Internet

The eggs dataset at Carnegie Mellon University's **StatLib** archive contains data on the yearly average number of eggs produced by female king crabs near Kodiak Island, Alaska.

```
filename kodiak
   url 'http://lib.stat.cmu.edu:80/crab/eggs';
   run;
data eggdata;
   infile kodiak;
   input year 1-2 numeggs 4-9;
   run;
```

kodiak is a nickname used by **SAS** to refer to the longer Internet address

Using File Transfer Protocol (FTP) to retrieve data.

```
filename kodiak ftp 'eggs' cd='/crab/'
      user='anonymous' pass='guest'
      host='lib.stat.cmu.edu';
data eggdata;
  infile kodiak;
  input year 1-2 numeggs 4-9;
  run;
```


SAS Datasets

SAS Data Set - a binary formatted representation of the input data set stored in such a way that future SAS programs do not need to input the data in again. SAS data sets contain **meta data** that tells the SAS program everything it needs to know about the variables in the data set.

Temporary SAS Data Sets - created and remain in working memory for the SAS session, but disappear when the SAS session ends. Fine for small to moderate size, simple input programs.

Permanent SAS Data Sets - created in one SAS session but stored on disk for later reuse. Convenient for large or complex input data sets that may require multiple analysis steps. Reduces time and computer resources.

LIBNAME statement - identifies to the SAS program where the previously created SAS data set is locted.

Temporary and Permanent SAS Data Sets

```
DATA myfile;
```

Creates a temporary data set that disappears at the end of the SAS session.

Does not creates any data set, simply reads in data from another SAS file or external data file and PUTs it to some other FILE. Nothing remains at the end of this data step except what you have filed.

```
DATA _NULL_ ;
```

```
LIBNAME mydir "c:\mydir";
DATA mydir.myfile;
```

Creates a permanent SAS (binary encoded) data set that is stored in the 'mydir' as myfile.SD2. Once created you never have to read this into SAS again. You can directly refer to it in other programs.

```
LIBNAME TOM "c:\TOM";
DATA TOM.TOM;
```

```
LIBNAME mydir "c:\mydir";
PROC SORT data=mydir.myfile;
```

Creating a Permanent Data Set

If a permanent data set is created, store it here.

```
LIBNAME college 'a:';
DATA original;
INPUT dept $ 1-8 count 10-13 class $ 15-21;
DATALINES;
FineArts  449 day
Science  1411 day
Music     259 evening
Language  759 day
;
DATA college.enrolled;
  SET original;
  IF class='evening' THEN DELETE;
  RUN;
PROC PRINT;
RUN;
```

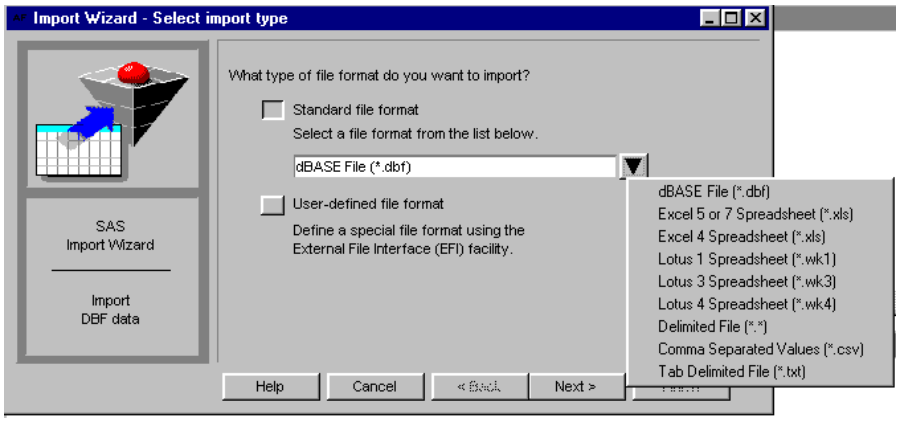
Make a permanent data set called ENROLLED and place it in the COLLEGE archive.

ENROLLED.SD2 stored on the **a** disk.

Create data set *name_1* from data set *name_2*

Reading Data from Other Applications

Data input and stored in special formats, such as those from spreadsheet applications, data base management systems can be input directly into SAS using the FILE -> IMPORT menu item and associated dialog box.



List Input

```
data grades;
input student $ quiz test project $ absences;
datalines;
Ann 84 90 A- 0
Bill    78   84 B 0
Cathy   95 89   A 1
David 84 88 B+ 1
;
```

One character variable, followed by two numeric variables followed by a character variable then ending with a numeric variable.

As simple as it gets.
Data columns are not lined up, but SAS doesn't care.

- At least one blank between variable data.
- No spaces in character data.
- Character data with less than or equal to 8 characters.

What if student is **Elizabeth** or **Mary Beth**?
What if **B+** is entered as **B Plus** or **B_Plus**?

Delimited Files

DELIMITER - character used to separate items.

```
Ann/84/90/A-/0
Bill/78/84/B/0
Cathy/95/89/A/1
David/84/88/B+/1
```

grades.txt data set on a disk.

Tell SAS what separates variables in the data set.

Alternate format `dlm=`

```
data grades;
  infile 'a:\grades.txt' delimiter='/';
  input name $ quiz test project $ absences;
  run;
```

If the file delimiter is a tab - e.g. from EXCEL or LOTUS Tab-delimited file.

```
infile 'a:\grades.txt' expandtabs;
```

Column Pointers

```
1234567890123456789
Ann   84 90 A- 0
Bill  78 84 B  0
Cathy 95 89 A  1
David 84 88 B+ 1
```

```
data grades;
infile 'a:\grades.txt' firstobs=2;
input @1 name $ @13 project $;
run;
```

"start at 13"

With column pointers you can tell SAS directly which column to begin reading a variable from.

- Go directly to the information you really need.
- Skip unnecessary information.
- Efficient data entry.

Writing nice programs

```
Data students;
 input name $ age degree $;
 datalines;
Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
proc print data=students;
run;
```

```
DATA Students;
INPUT Name $ Age Degree $;CARDS;

Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
PROC PRINT Data=Students; Run;
```

Rule: Try to keep at most one SAS statement to a line.
Rule: Indent subcommands within DATA and PROC statements.

Comments

```
*This comment statement starts with an asterick;

/* This is also a comment statement. */

/*
  This is a comment block. It can be used
  to provide more detailed comments. Also
  blocks of code can be commented out
  when you are done running that part of the
  program.  Note, anything, including special
  characters !@#$%^&() except asterick and
  forward slash can be used.
*/
```


Using Block Comments

Block out some old code but keep it in the program as a record of what you have attempted.

```
Data students;
 input name $ age degree $;
 datalines;
Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
/*
proc print data=students;
  var name age;
  run;
*/
proc sort data=students;
  by degree name;
  run;
proc print data=students;
  by degree;
  run;
```

Variable names

```
Data students;
 input A $ B C $;
 datalines;
Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
proc print data=students;
run;
```

```
Data students;
 input first_name $ age
     degree_sought $;
 datalines;
Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
proc print data=students;
run;
```

You have up to 32 characters for each variable name, with up to 37 possible characters in each position except the first which has 27. Use this flexibility to give variables understandable mnemonics.

The OPTIONS statement

Place at the beginning of your program to control output options.

```
options linesize=80
        pagesize=54
        nocenter
        nodate
        nonumber;
```

linesize=124
to print wide output.

pagesize=500
to print data without page breakes.

The TITLE and FOOTNOTE statements

Use a TITLE and/or FOOTNOTE statement to place comment information at the top (TITLE) or bottom (FOOTNOTE) of each output page.

Once set, TITLE and FOOTNOTE information will be printed for each procedure output unless a new TITLE or FOOTNOTE statement is encountered.

```
title1 'First line';
title2 'Second line';
title3 'Third line';
footnote1 'Line one';
footnote2 'Line two';
```

```
Data students;
 input name $ age degree $;
 datalines;
Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
proc print data=students;
title 'Four New Post-Bacc Students';
footnote 'Ages as of last birthday';
run;
```


Labels

```
data basket;
  input jerseyno $ name $ _3pmpo97 _3papo97 @@;
  label jerseyno='Jersey number'
        name='Name'
        _3pmpo97='3 pt. goals made, 1997 playoffs'
        _3papo97='3 pt. goals attempted, 1997 playoffs';
  datalines;
0  Irlbeck 4 19   00 Morrison 6 9    11 Perkins 1 4
13 Pinson 4 9     14 Camacho 0 0     22 Gragg 0 0
23 Mitchell 0 0   35 Garcia 0 1     42 Cannon 3 3
44 McGuire 0 0    51 Betts 2 4       55 Galloway 2 4
;
  run;
proc print data=basket label;
  title 'Playoff results';
  run;
```

Labels in Output

Playoff results

OBS	Jersey number	Name	3 pt. goals made, 1997 playoffs	3 pt. goals attempted, 1997 playoffs
1	0	Irlbeck	4	19
2	00	Morrison	6	9
3	11	Perkins	1	4

Label statements provide variable labels with more information.

Example Program

```
data finances;
  input @1 country $Char15. @16 educ_exp comma5. @21 gdp comma6.
        @27 ptratio;
  format educ_exp gdp dollar7.;
  label country  ='Country'
        educ_exp ='Expenditures'
        gdp      ='Domestic Product'
        ptratio  ='Pupil/Teacher';
datalines;
Canada         4,05417,35516.2
United States  3,39818,29718.4
United Kingdom 2,47412,52915.7
Australia      2,06013,52315.7
;
  run;
proc print data=finances;
  run;
proc contents data=finances;
  run;
```


Proc Contents

```
CONTENTS PROCEDURE

Data Set Name: WORK.FINANCES
Observations:          4
Member Type:           DATA
Variables:             4
Engine:                V612
Indexes:               0
Created:               9:22 Tue, Dec 16, 1997
Observation Length:    39
Last Modified:         9:22 Tue, Dec 16, 1997
Deleted Observations:  0
Protection:
Compressed:            NO
Data Set Type:
Sorted:                NO
Label:
```

No of observations

No of variables

SAS version

Is dataset sorted?

Proc Contents

```
-----Engine/Host Dependent Information-----

Data Set Page Size:          8192
Number of Data Set Pages:    1
File Format:                 607
First Data Page:             1
Max Obs per Page:            208
Obs in First Data Page:      4

    -----Alphabetic List of Variables and
Attributes-----

#   Variable   Type   Len   Pos   Format     Label
1   COUNTRY    Char    15     0              Country
2   EDUC_EXP   Num      8    15   DOLLAR7.   Expenditures
3   GDP        Num      8    23   DOLLAR7.   Domestic Product
4   PTRATIO    Num      8    31              Pupil/Teacher
```

You may need this if you later need to move data to a different operating system.

Formats and Labels

PROC PRINT options

DATA =*name*	- specifies data set name
DOUBLE	- double-spaces output
NOOBS	- suppresses the observation number in the first column
HEADING=H	- column headings should be printed (h)orizontally
HEADING=V	- column headings should be printed (v)ertically
UNIFORM	- asks for all pages of output to have the same appearance

```
proc print data=basket(obs=5);
  var jerseyno;
  id name;
  run;
```

Print only the first 5 observations for variable **jerseyno** but add the identification data in variable **name**.

Creating your own formats - PROC FORMAT

```
proc format;
  value $sexfmt 'F'='Female' 'M'='Male';
  value ratefmt 0-<100='Low'
                100-HIGH='High'
                      .='Missing';
  run;
```

```
proc print data=kids noobs label;
  var firstnam lastname gender wklyrate;
  format gender $sexfmt. wklyrate ratefmt.;
  title 'Day care roster';
  run;
```

NOTE: Formats always end with a period.

Formatted Output

```
Day care roster

First name     Last name      Gender    Rate

Douglas        Lindgren       Male      High
Elizabeth      Wilkerson      Female    Low
Evangeline     Chambers       Female    High
Arthur         Hollander      Male      Missing
Christopher    Kalbfleisch    Male      High
Stacy          Siegel         Female    High
```

```
First name   Last name             Birthday   Gender      Rate
Douglas      Lindgren       August 29, 1996     M       $115.00
Elizabeth    Wilkerson     January 13, 1997     F        $90.00
Evangeline   Chambers        March 11, 1997     F       $100.00
Arthur       Hollander        July 19, 1996     M             .
Christopher  Kalbfleisch     April 13, 1995     M       $115.00
Stacy        Siegel       November 15, 1996     F       $100.00
```


PROC SORT

Sorting data sets is probably the most common and useful procedure applied.

```
proc sort data=kids;
  by gender birthday;
  run;
proc print data=kids noobs label;
  var firstnam lastname birthday wklyrate;
  by gender;
  format birthday worddate18. wklyrate dollar7.2;
  title 'Day care roster';
  run;
```

Anything done with a BY statement requires the data be SORTED first.

Sorted Output

Sorted by birthday within Gender.

```
Day care roster

Gender=F

First name     Last name              Birthday       Rate

Stacy          Siegel         November 15, 1996     $100.00
Elizabeth      Wilkerson       January 13, 1997      $95.00
Evangeline     Chambers          March 11, 1997     $100.00


Gender=M

First name       Last name                Birthday       Rate

Christopher      Kalbfleisch          April 13, 1995     $115.00
Arthur           Hollander             July 19, 1996          .
Douglas          Lindgren            August 29, 1996     $115.00
```


Missing Data

"No Data"

```
Ann    84 90 A- 0
Bill   78 84 B  0
Cathy  95 89 A  1
David  84 88 B+ 1
Eve    79       4
```

```
/*
 Lesson 3 Special Types of Data and Working with Spreadsheets
  Example SAS Programs */
/*
  MISSING DATA
*/
* Column Input ;
options nocenter nodate ;
data grades;
  infile 'c:\portier\teaching\sta5106_F00\lesson3\grades.txt';
  input name $ 1-5 quiz 7-8 exam 10-11 project $ 13-14 absences 16;
run;
proc print;
  title 'Using column input to handle missing data';
  run;
```


Output

```
Using column input to handle missing data          1

OBS    NAME     QUIZ    EXAM    PROJECT    ABSENCES

 1     Ann       84      90       A-           0
 2     Bill      78      84       B            0
 3     Cathy     95      89       A            1
 4     David     84      88       B+           1
 5     Eve       79       .                    4
```

Missing numeric data indicated by period.

Missing character data indicated by nothing (blank space).

Missing Data Codes

```
/*
Putting in a special code for missing data.
*/
data grades;
  input name $ quiz exam project $ absences;
  datalines;
Ann   84 90 A- 0
Bill  78 84 B  0
Cathy 95 89 A  1
David 84 88 B+ 1
Eve   79 .  ?  4
;
run;
Proc print;
  title "Using special characters to handle missing data";
run;
```

Calculating New Variables

New variables are created from input data values using standard algebraic expressions and mathematical functions.

```
Newvar = var1 + var2 ;   addition
Newvar = var1 - var2;    subtraction
Newvar = var1 * var2;    multiplication
Newvar = var1 / var2;    division
Newvar = var1 ** var2;   exponentiation  (var1^var2)

Order of operation can be controlled by parenthesis.

Newvar = (var1 + var2) ** (var3 / (var4*var5)) - var6 ;
```

```
Gigabyte = harddriv/1e9 ;
kilobaud = modem / 1000;
perimeter = (2*(3/5)*monitor + 2*(4/5)*monitor)*2.54;
area = ((((3/5)*monitor)*((4/5)*monitor))/2)*(2.54**2);
```


Mathematical Functions

Newvar = log(var1);	value is natural logarithm of var1
Newvar = log10(var1);	value is common logarithm of var1
Newvar = log2(var1);	value is base 2 logarithm of var1
Newvar = sqrt(var1);	value is square root of var1
Newvar = mdy(month, day, year);	create SAS date variable from individual month, day and year values.
Newvar = abs(var1);	value is the absolute value of var1

See tables in textbook.

Logical Variables

IF logical_condition_true THEN action_1;
ELSE action_2;

Logical_condition	A SAS statement containing a logical operator.
Logical operators	EQ equals (=)
	NE not equal (~=, ^=)
	GT greater than (>)
	LT less than (<)
	GE greater than or equal to (>=)
	LE less than or equal to (<=)
	AND all comparisons must be true (&)
	OR only one comparison must be true (!, |)
Actions	ANY SAS data or macro statement (e.g. another assignment, IF, DO …)

RECOMMENDATION: Use logical operators EQ, LT, LE, GT, GE, AND, OR

```
data computer;
  input @1 vendor $16. @17 harddriv modem monitor price;
/* A customer wants to avoid DogBytes. */
  if vendor EQ 'DogBytes' then avoid=1;
    else avoid=0;
/*                                          Logical operator.
  Equivalently, use the following:          Assignment operator.

  if vendor NE 'DogBytes' then avoid=0;
    else avoid=1;
*/
/* The customer wants lots of disk space. */
  if harddriv LE 2e9 then diskspac=1;
    else if 2e9 LT harddriv LE 4e9 then diskspac=2;
       else if harddriv GT 4e9 then diskspac=3;
/*Check the price range and monitor size simultaneously.*/
  if (price LE 1600 AND monitor GE 15) then goodbuy='Yes';
    else goodbuy='No ';
```

Why use logical operators?

1) To remove observations or otherwise subset the data set.

```
data cheap;
  set computer;
  if price LE 1600;
```

2) To differentially assign values.

```
if age LE 21
  then age_grp=1;
  else
    if 21 LT age LE 55
      then age_grp=2;
      else
        if age GT 55
          then age_grp=3;
```


Removing (DROPing) Variables

KEEP Tells SAS exactly which variables to keep in the data set.
DROP Tells SAS exactly which variables to drop before storing the data set.

```
data costdata;
  set computer;
  keep vendor price;
run;
```

```
data costdata;
  set computer;
  drop harddriv
       modem
       monitor;
run;
```

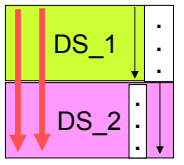
RECOMMENDATION: Use DROP unless the data set has a lot of variables and you only wish to work with a couple.

Combining Data Sets

Red 6.1, 6.2
Purple 5.1, 5.2

Combining Data Sets

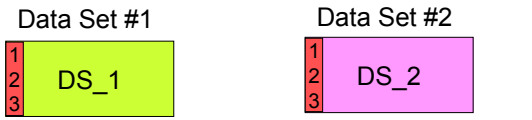
SET **Stack data**
Common variables




```
DATA new_ds;
  SET ds_1 ds_2 ;
  RUN;
```

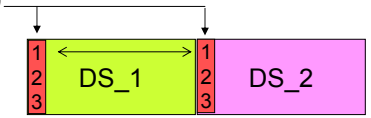
If a variable exists only in one of the data sets it will be assigned as missing for part of the final data set.

Combining Data Sets



MERGE **Add variables**
New variables
Common ID

Data sets must be sorted in proper order.




```
DATA new_ds;
  MERGE ds_1 ds_2 ;
  BY com_var ;
  RUN;
```



```
data rain95;
  input month rainfall @@;
  year=1995;
  datalines;
 1 3.08 2 1.07 3 6.14 4 5.18 5 2.47 6 7.55
 7 7.66 8 7.20 9 2.10 10 4.33 11 3.15 12 1.29
;
  run;
data rain96;
  input month rainfall @@;
  year=1996;
  datalines;
 1 0.97 2 0.66 3 10.52 4 1.72 5 2.01
 6 6.05 7 11.0 8 4.90 9 2.23 10 6.18
 11 1.73 12 6.63
;
  run;
data rain9596;
  set rain95 rain96;
  run;
```

```
1 1 3.08 1995
2 2 1.07 1995
```

rain95

```
1 1 0.97 1996
2 2 0.66 1996
```

rain96

```
1 1 3.08 1995
2 2 1.07 1995
3 1 0.97 1996
4 2 0.66 1996
```

rain9596


```
data rain9596;
 set rain95(in=i95) rain96(in=i96);
 if (i95=1 & i96=0)
  then year=1995;
  else
    if (i95=0 & i96=1)
     then year=1996;
run;
```

(IN=I95) option on the SET statement allows the creation of a new variable, I95, which is equal to 1 for just those observations that come from data set rain95, zero otherwise. The variable I96 is likewise defined for rain96. The IF then statements are used to assign values to year dependent on the data set the rainfall observations came from.


```
data rain95;
  input month rainfall @@;
  datalines;
 1 3.08 2 1.07 3 6.14 4 5.18 5 2.47 6 7.55
 7 7.66 8 7.20 9 2.10 10 4.33 11 3.15 12 1.29
;
  run;
proc sort data=rain95;
  by month;
  run;
data rain96;
  input month rainfall @@;
  datalines;
 1 0.97 2 0.66 3 10.52 4 1.72 5 2.01 6 6.05 7 11.00
 8 4.90 9 2.23 10 6.18 11 1.73 12 6.63
;
proc sort data=rain96;
  by month;
  run;
data rain9596;
  merge rain95(rename=(rainfall=r95))
        rain96(rename=(rainfall=r96));
  by month;
  run;
```

In this case, when we merge the two data sets we wish to have each years rainfall in a separate column.

This requires renaming the rainfall variable at the time of merging.

```
data rain95;
  input month r95 @@;
  datalines;
 1 3.08 2 1.07 3 6.14 4 5.18 5 2.47 6 7.55
 7 7.66 8 7.20 9 2.10 10 4.33 11 3.15 12 1.29
;
  run;
proc sort data=rain95;
  by month;
  run;
data rain96;
  input month r96 @@;
  datalines;
 1 0.97 2 0.66 3 10.52 4 1.72 5 2.01 6 6.05 7 11.00
 8 4.90 9 2.23 10 6.18 11 1.73 12 6.63
;
proc sort data=rain96;
  by month;
  run;
data rain9596;
  merge rain95 rain96 ;
  by month;
  run;
```

Simpler solution is to name the rainfall values in each file differently at creation.

OBS	MONTH	R95	R96
1	1	3.08	0.97
2	2	1.07	0.66
3	3	6.14	10.52
4	4	5.18	1.72
5	5	2.47	2.01
6	6	7.55	6.05
7	7	7.66	11.00
8	8	7.20	4.90
9	9	2.10	2.23
10	10	4.33	6.18
11	11	3.15	1.73
12	12	1.29	6.63

MACROs

A SAS macro is a short "program within a program" which can expedite certain data handling tasks. Like arrays, they are useful for performing repetitive procedures. Writing macros can be very complicated.

```
data credit;
  input name $ card_no debt;
  datalines;
Stephens 67382910 516.32
Hill     74839222  43.99
Halford  74283848  24.12
Myatt    82843054  88.65
Guerrant 37759831  29.54
;
  proc sort data=credit;
    by name;
    run;
  proc print data=credit;
    format debt dollar7.2;
    run;
  proc sort data=credit;
    by dept;
    run;
  proc print data=credit;
    format debt dollar7.2;
    run;
  proc sort data=credit;
    by card_no;
    run;
  proc print data=credit;
    format debt dollar7.2;
    run;
```

Here, the same processes are repeated three times, only the variable in the BY statement changes.

Simple Macro

```
%macro arrange(v=,dat=);
  proc sort data=&dat;
    by &v;
    run;
  proc print data=&dat;
    format debt dollar7.2;
    run;
%mend arrange;
```

```
data credit;
  input name $ card_no debt;
  datalines;
Stephens 67382910 516.32
Hill     74839222  43.99
Halford  74283848  24.12
Myatt    82843054  88.65
Guerrant 37759831  29.54
;
```

```
%arrange(v=name,dat=credit);
%arrange(v=debt,dat=credit);
%arrange(v=card_no,dat=credit);
```

The macro called arrange needs to have two inputs specified: the dataset (dat=) and the variable to sort on (v=). Within the macro, the values you provide for inputs will be referred to as &dat and &v.

After the arrange macro has been defined, it can be called any number of times with different inputs.

```
%arrange(v=name,dat=credit);
```

OBS	NAME	CARD_NO	DEBT
1	Guerrant	37759831	$29.54
2	Halford	74283848	$24.12
3	Hill	74839222	$43.99
4	Myatt	82843054	$88.65
5	Stephens	67382910	$516.32

```
%arrange(v=debt,dat=credit);
```

OBS	NAME	CARD_NO	DEBT
1	Halford	74283848	$24.12
2	Guerrant	37759831	$29.54
3	Hill	74839222	$43.99
4	Myatt	82843054	$88.65
5	Stephens	67382910	$516.32

```
%arrange(v=card_no,dat=credit);
```

OBS	NAME	CARD_NO	DEBT
1	Guerrant	37759831	$29.54
2	Stephens	67382910	$516.32
3	Halford	74283848	$24.12
4	Hill	74839222	$43.99
5	Myatt	82843054	$88.65

Proc Capability

SAS/QC Software Module

```
PROC CAPABILITY < options >;
VAR variables;
SPEC <options >;
CDFPLOT <variables > < / options >;
COMPHISTOGRAM <variables > /
CLASS=(class-variables) <options >;
HISTOGRAM <variables > < / options >;
PPPLOT <variables > < / options >;
PROBPLOT <variables > < / options >;
QQPLOT <variables > < / options >;
INSET keyword-list < / options >;
INTERVALS <variables > < / options >;
OUTPUT <OUT=SAS-data-set> keyword=names <
... keyword=names >;
```



```
proc mixed data=chahoud.prostate ratio covtest;
 by aged ;
 class trt dam day0;
 model prostatewt = trt /outpred=four ;
 random day0*dam(trt);
 lsmeans trt / adjust=dunnett diff=control('control');
 run;
proc capability data=four normal;
 var resid  ;
 ppplot resid / square ;
 run;
```


The CAPABILITY Procedure
Variable: Resid

Moments

N	237	Sum Weights	2
Mean	1.0189E-16	Sum Observations	2.4147E-
Std Deviation	0.05828098	Variance	0.003396
Skewness	0.19010952	Kurtosis	1.677587
Uncorrected SS	0.80161479	Corrected SS	0.801614
Coeff Variation	5.72013E16	Std Error Mean	0.003785

Basic Statistical Measures

Location		Variability	
Mean	0.00000	Std Deviation	0.05828
Median	-0.00007	Variance	0.00340
Mode	.	Range	0.46050
		Interquartile Range	0.07383

Tests for Location: Mu0=0

Test	-Statistic-		-----p Value------	
Student's t	t	2.69E-14	Pr > |t|	1.0000
Sign	M	-0.5	Pr >= |M|	1.0000
Signed Rank	S	-421.5	Pr >= |S|	0.6908

Tests for Normality

Test	--Statistic---		-----p Value-----	
Shapiro-Wilk	W	0.979522	Pr < W	0.002
Kolmogorov-Smirnov	D	0.052619	Pr > D	0.107
Cramer-von Mises	W-Sq	0.092381	Pr > W-Sq	0.144
Anderson-Darling	A-Sq	0.811305	Pr > A-Sq	0.037

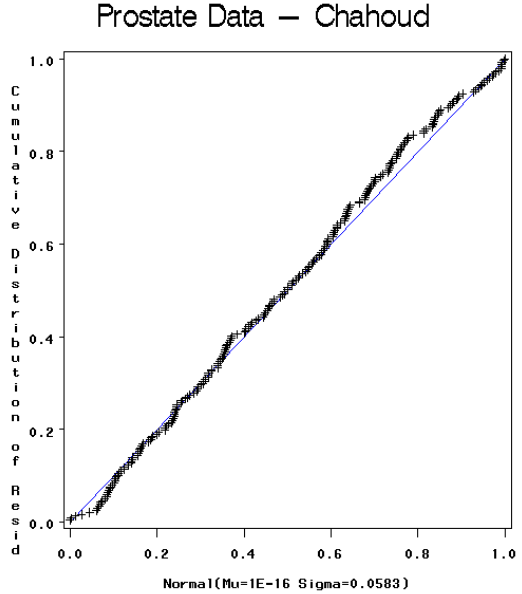
The CAPABILITY Procedure
Variable: Resid

Quantiles (Definition 5)

Quantile	Estimate
100% Max	2.25058182E-01
99%	1.47631596E-01
95%	1.01365275E-01
90%	6.82945517E-02
75% Q3	3.37336817E-02
50% Median	-7.26981662E-05
25% Q1	-4.00992020E-02
10%	-7.15107723E-02
5%	-8.16682086E-02
1%	-1.32496587E-01
0% Min	-2.35439885E-01

Extreme Observations

Lowest		Highest	
Value	Obs	Value	Obs
-0.2354398847	131	0.141002628	111
-0.1722112518	174	0.142569163	139
-0.1324965870	180	0.147631596	150
-0.1134308195	140	0.165956019	232
-0.0991156732	103	0.225058182	67




```
proc capability data=hotdrink;
  var amount;
  by beverage;
  histogram amount / normal;
  run;
```

beverage=Coffee

The CAPABILITY Procedure
Fitted Normal Distribution for amount

Parameters for Normal Distribution

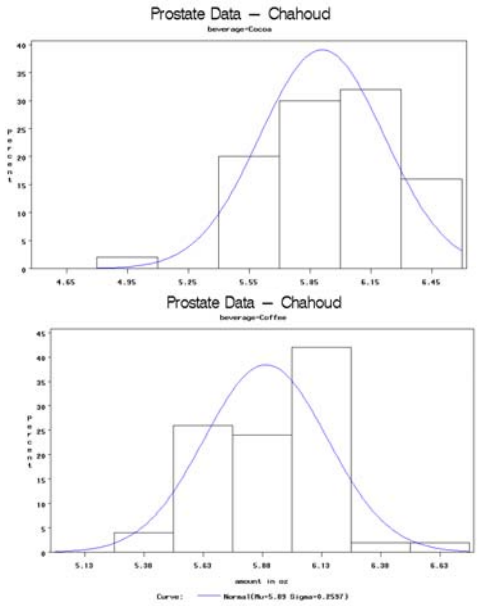
Parameter	Symbol	Estimate
Mean	Mu	5.89
Std Dev	Sigma	0.259709

Goodness-of-Fit Tests for Normal Distribution

Test	Statistic		DF	p Value	
Kolmogorov-Smirnov	D	0.1240531		Pr > D	0.053
Cramer-von Mises	W-Sq	0.1058638		Pr > W-Sq	0.093
Anderson-Darling	A-Sq	0.6185684		Pr > A-Sq	0.102
Chi-Square	Chi-Sq	10.9324220	3	Pr > Chi-Sq	0.012

Quantiles for Normal Distribution

Percent	Quantile Observed	Quantile Estimated
1.0	5.30000	5.28583
5.0	5.50000	5.46282
10.0	5.60000	5.55717
25.0	5.70000	5.71483
50.0	5.90000	5.89000
75.0	6.10000	6.06517
90.0	6.20000	6.22283
95.0	6.20000	6.31718
99.0	6.60000	6.49417



Outputting Summary Statistics

Often, you will want to compute summary statistics (typically the mean, median or standard deviation) for your data and use these summary statistics as inputs to a more formal analysis.

Typical example - multiple sub-samples have been measured for each experimental unit and we are not particularly interested in the sub-sample to sub-sample variability. In this case we would first run a PROC UNIVARIATE or PROC MEANS on an appropriately sorted data set to output experimental unit means.

NOPRINT and OUTPUT

```
PROC SORT DATA=hotdrink;
  BY beverage;
  RUN;
PROC UNIVARIATE NOPRINT DATA=hotdrink;
  VAR amount;
  BY beverage;
  OUTPUT OUT=results N=number MEAN=avg_amt  STD=std_dev;
  RUN;
PROC PRINT DATA=results;
  RUN;
```

RESULT DATA SET

OBS	BEVERAGE	NUMBER	AVG_AMT	STD_DEV
1	Cocoa	50	5.91	0.30656
2	Coffee	50	5.89	0.25971

PLOTS AND CHARTS

- PROC INSIGHT
- PROC PLOT
- PROC GPLOT
- PROC GCHART
- PROC G3D
- PROC GCONTOUR
- PROC GMAP
- PROC GCONTOUR

SAS/GRAPH

Types of Plots

Box Plots

Maps

Scatterplots

3-D Plots

Charts

Time Series Plots

Low Resolution Plots

Hi-Resolution Plots

Medium-Resolution Plots

PROC PLOT

The low-resolution plots in SAS are adequate for simple visualization and model checking. PROC PLOT is the basic SAS procedure for producing two-way scatterplots of X and Y variables. It can be modified to show levels of a third variable Z, to plot two sets of data on the same axes, and put multiple plots on one page.

Example

Dimensions of the carapaces (shells) of female painted turtles, measured in millimeters.

```
DATA females;
  INPUT length width height @@;
  DATALINES;
 98  81 38 103  84 38 103  86 42 105  86 42 109  88 44
123  92 50 123  95 46 133  99 51 133 102 51 133 102 51
134 100 48 136 102 49 138  98 51 138  99 51 141 105 53
147 108 57 149 107 55 153 107 56 155 115 63 155 117 60
158 115 62 159 118 63 162 124 61 177 132 67
;
  RUN;
PROC PLOT DATA=females;
  PLOT width*length;
  RUN;
```



```
Plot of WIDTH*LENGTH.  Legend: A = 1 obs, B = 2 obs, etc.
 140 +
     |                                                  A
     |
     |                                              A
 120 +                                            A
     |                                        B A
WIDTH |
     |                                   AA A
     |                              B A  A
 100 +                              AA B
     |                         A
     |                         A
     |              AA A
     |              A
  80 +          A
     +-----------+----------+----------+----------+----------+-
     80         100        120        140        160        180

                    LENGTH
```

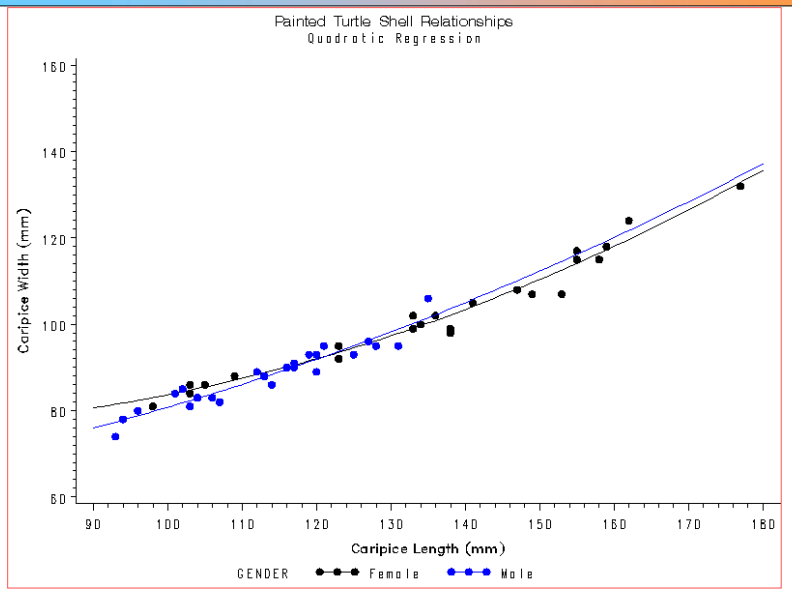
Low resolution plot

SAS/GRAPH

High Resolution

Printing & Displaying	Mapping	Graphic Utilities
GANNO	GMAP	GDEVICE
GPRINT	GPROJECT	GFONT
GREPLAY	GREDUCE	GIMPORT
GSLIDE	GREMOVE	GKEYMAP
		GOPTIONS
Charting & Plotting	Advanced Mapping	GTESTIT
GCHART	SAS/GIS	
GPLOT	Fonts	Multimedia
ANNOTATE=	Project Management	Quality Control
Graphics Statements & Windows	Decision Analysis	(only w/ SAS/QC)
Graphics Devices & Drivers	(only w/ SAS/OR)	CAPABILITY
	DTREE	CUSUM
3-Dimensional Plotting	GANTT	ISHIKAWA
GCONTOUR	NETDRAW	MACONTROL
G3D	PROJMAN Menu	PARETO
G3GRID	System	SHEWHART

```
/* Set up graphics parameters and output file for Profile Plot */

filename gsasfile "e:/portier/teaching\sta5106_00\lesson8\turtle.gif";
goptions reset=all gaccess=gsasfile autofeed dev=gif ;
/* Establish symbols and interpolation for the plot */

symbol1 i=RQ v=dot c=black;
symbol2 i=RQ v=dot c=blue ;

/* Establish legend options and axes information */

legend1 across=2;
axis1 label=(a=90 h=1.2 f=duplex 'Caripice Width (mm)') order=(60 TO 160 BY 20);
axis2 offset=(2) label=(h=1.2 f=duplex 'Caripice Length (mm)');
title1 J=C h=1.2 'Painted Turtle Shell Relationships';
title2 j=C 'Quadratic Regression';

proc gplot data=all;
  plot width*length=gender /
    vaxis=axis1 haxis=axis2 legend=legend1;
  run;
```

PROC INSIGHT

SAS/INSIGHT software is an interactive tool for data exploration and analysis. With it you can explore data through a variety of interactive graphs and analyses linked across multiple windows. You can analyze univariate distributions, investigate multivariate distributions, and fit explanatory models using analysis of variance, regression, and the generalized linear model.

```
PROC INSIGHT DATA=mydata;
RUN;
```