Introduction to SAS

SAS (https://en.wikipedia.org/wiki/SAS_(software)) was developed in the early 1970s at North Carolina State University. It is originally intended for management and analysis of agricultural field experiments and now it is the most widely used statistical software. SAS is used to stand for "Statistical Analysis System", but now it is not an acronym for anything. SAS is pronounced "sass", not spelled out as three letters.

SAS is the most widely used statistical software due to the following several reasons:

- It is often better with very large datasets and memory;
- It can deal with multiple datasets at the same time.
- It is better for data manipulation.
- It is better on the Job Market.

The SAS software suite has more than 200 components (https://support.sas.com/software/). Some of the SAS components include:

- Base SAS – Basic procedures and data management
- SAS/STAT – Statistical analysis
- SAS/IML – Interactive matrix language

We will use Base SAS and SAS/STAT in this course to manipulate data and conduct basic statistical data analysis. SAS/IML will be used a little bit in our course.

SAS/IML (https://support.sas.com/) software, is like statistical software R, gives you access to a powerful and flexible programming language (Interactive Matrix Language) in a dynamic, interactive environment. The fundamental object of the language is a data matrix. You can use SAS/IML software interactively (at the statement level) to see results immediately, or you can store statements in a module and execute them later. The programming is dynamic because necessary activities such as memory allocation and dimensioning of matrices are done automatically. SAS/IML software is of interest to users of SAS/STAT software because it enables you to program your methods in the SAS System.

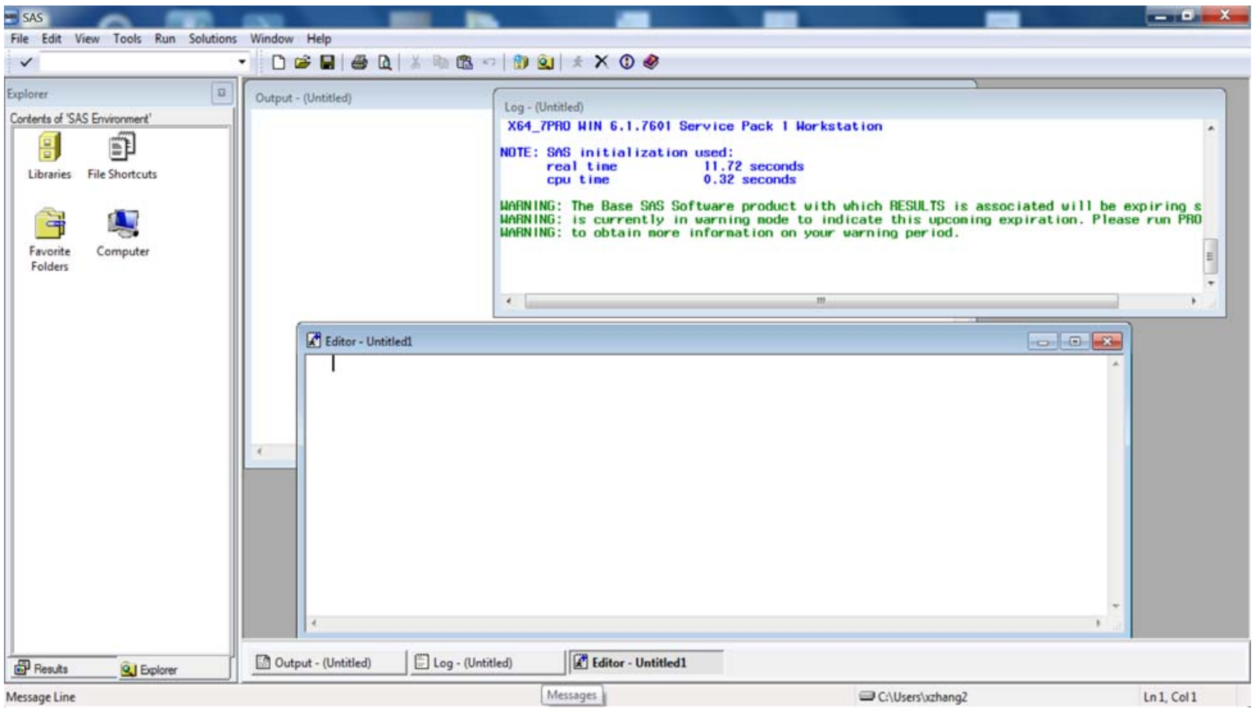
R is a matrix-based programming language that allows you to program statistical methods reasonably quickly. It's open source software, and many add-on packages for R have emerged, providing statisticians with convenient access to new research. Many new statistical methods are first programmed in R. SAS/IML Studio (https://support.sas.com/rnd/app/studio/) also provides the capability to interface with the R language. R will be used in our course MATH 402- Applied Statistical Methods.

SAS University Edition: A single-user license of the basic windows SAS Analytics package can be several thousand dollars. The good news is that SAS University Edition (https://www.sas.com/en_us/software/university-edition.html#) is free for you to use from your PC, Mac or Linux. The University Edition features Base SAS, SAS/STAT, SAS/IML, SAS/ACCESS, and SAS Studio (https://www.sas.com/en_us/offers/14q1/122603-sas-for-academia/overview.html). SAS works through virtualization software via your browser in standalone mode once you download it to your PC, Mac or Linux workstation.

SAS Windows: How do we download and install the SAS University Edition? Because SAS University Edition is a virtual application (or vApp), you need virtualization software to run it. There will be instructions and a list of steps to follow on this webpage: https://www.sas.com/en_us/software/university-edition/download-software.html. For example, or windows users, you can follow this Installation Guide

http://support.sas.com/software/products/university-edition/docs/en/SASUniversityEditionInstallGuideWindows.pdf

The following is the interface of SAS 9.4 PC version.



```
 X64_7PRO WIN 6.1.7601 Service Pack 1 Workstation

NOTE: SAS initialization used:
      real time           11.72 seconds
      cpu time            0.32 seconds

WARNING: The Base SAS Software product with which RESULTS is associated will be expiring s
WARNING: is currently in warning mode to indicate this upcoming expiration. Please run PRO
WARNING: to obtain more information on your warning period.
```

The SAS System for Windows consists of the 4 main windows:

1. Results and Explorer window

a. Explorer pane is browsing tool for SAS libraries

b. Results pane shows a tree-like summary of the output window. You can select, delete, or edit the output before printing, saving, or copying the results

2. Enhanced Program Editor window – Used to create, edit and execute SAS programs

3. Output Window – Displays output from SAS program. Can view, print, copy or

save information in the output window

4. Log Window – Reports on progress of SAS procedures (BLUE). Displays error messages (RED) and warnings (GREEN).

General Information

- All statement lines must end with a semi-colon.
- Comments are indicated in 2 ways.

  1. Start line with * and end with a semi-colon

  2. Enclose with as /* put comment text here */

- End all procedures with RUN.

To start our first SAS program, we need to get data into SAS first. There are several ways to get data into SAS: 1) Read in-stream data, 2) Use INFILE statement, and 3) Import the data from Excel. Let's use several examples to show how to read raw data into SAS.

Problem.

Reports of results from clinical trials often include statistics about "baseline characteristics", so we can see that different groups have the same basic characteristics. It is of interest to see the difference between the mean age of men (μ1) and the mean age of women (μ2). A sample of 40 men and a sample of 40 women are randomly and independently selected and the ages are recorded in the files MandFBODY.txt and MandFBODY.xls.

Method 1(Read in-stream data or copy the data to the SAS code):
One of the most common ways to read data into SAS is by reading the data instream in a **data step** - that is, by **typing the data directly** into the syntax of your SAS program. This approach is good for relatively small datasets. Spaces are usually used to "delimit" (or separate) free formatted data.

```
data age1;
input male female@@;
cards; /*or use datalines;*/
18     60
20     24
43     49
39     62
60     53
18     18
57     41
27     21
20     21
18     19
63     19
20     58
24     44
46     52
29     48
63     36
21     48
45     34
40     22
50     61
48     21
64     33
18     32
50     37
20     19
20     51
47     35
19     18
55     60
23     58
21     60
19     48
64     31
30     29
43     46
23     18
64     50
40     20
23     56
44     18
;
run;
```

After reading in the data with a data step, it is usually a good idea to print the first few (or all) cases of your dataset to check that things were read correctly. For example, we print the first 10 cases of the data set.

```
TITLE "Age data";
proc print data= age1 (obs=10);
run;
```

Method 2(use INFILE statement in the data step):

```
data age2;
infile "E:\SASHandout\MandFBODY.txt" firstobs=2; /*change the directory if
necessary*/
input male female;
run;

TITLE "Age data";
proc print data= age2 (obs=10);
run;
```

Method 3 (use IMPORT procedure):

```
proc import
datafile=" E:\SASHandout\MandFBODY.xls"
out=age3 dbms=xls replace;
getnames=yes;
run;

TITLE "Age data";
proc print data= age3 (obs=10);
run;
```

The SAS Import wizard can be used to access spreadsheets (Excel, Lotus) and database (Access) files as well. It is most convenient if the variable names are in the first line of the Excel spreadsheet and comply with SAS naming conventions – no more than 8 characters long, no spaces in the middle, and start with a letter.   Go to "File" -> "Import Data..." … We do not use this method in this course.

For more information, please go to http://statistics.ats.ucla.edu/stat/sas/modules/input.htm

Variable Transformations:
Transforming data is an important technique in exploratory data analysis. For example, centering and scaling are simple examples of transforming data. There is no need to transform your raw data outside of SAS. You can use Data step to transform your data.

```
data data1;
  input x y gender$ @@;
  cards;
4.5   17.5 M
2.3   20.5 F
6.7   13.6 F
8.4   17.8 M
-2.0  14.7 M
;
run;

proc print data=data1;
run;
```

Now add a variable z, a log transformation of y, to the data set.

```
data data2;
  input x y gender$ @@;
  z=log(y);
  cards;
4.5   17.5 M
2.3   20.5 F
6.7   13.6 F
8.4   17.8 M
-2.0  14.7 M
;
run;
```

Equivalently, we can do this as well.

```
data data3;
set data1;  /* create data3 from data1 */
z=log(y);
run;
```

Deleting rows: We can delete some rows based on a control condition.

```
data data4;
set data3;
   if x > 0;  /*  keep rows with x>0 only */
run;

proc print data=data4;
run;
```

Deleting variables: We can delete a column as well.

```
data data5;
set data4;
   drop y;  /*  remove variable y*/
run;
```

'do … end' loop: It is similar as the "for" loop in R which is used for iterating over a sequence.

```
data uniform;
   do i = 1 to 10;
      x = ranuni(0); /* random uniform number between 0 & 1 */
      output;
   end;

proc print data=uniform;
run;
```

Note the use of a do loop, which is ended by an end; phrase. The output forces creation of a new case for each uniform number. Each case in set a will have the variables x and i. Here is a list of random number generators:

```
   x = ranuni(seed)           /* uniform between 0 & 1 */
   x = a+(b-a)*ranuni(seed);  /* uniform between a & b */
   x = ranbin(seed,n,p);      /* binomial size n prob p */
   x = rancau(seed);          /* cauchy with loc 0 & scale 1 */
   x = a+b*rancau(seed);      /* cauchy with loc a & scale b */
   x = ranexp(seed);          /* exponential with scale 1 */
   x = ranexp(seed) / a;      /* exponential with scale a */
   x = a-b*log(ranexp(seed)); /* extreme value loc a & scale b */
   x = rangam(seed,a);        /* gamma with shape a */
   x = b*rangam(seed,a);      /* gamma with shape a & scale b */
   x = 2*rangam(seed,a);      /* chi-square with d.f. = 2*a */
   x = rannor(seed);          /* normal with mean 0 & SD 1 */
   x = a+b*rannor(seed);      /* normal with mean a & SD b */
   x = ranpoi(seed,a);        /* poisson with mean a */
   x = rantri(seed,a);        /* triangular with peak at a */
   x = rantbl(seed,p1,p2,p3); /* random from (1,2,3) with probs p1,p2,p3*/
```

T Test

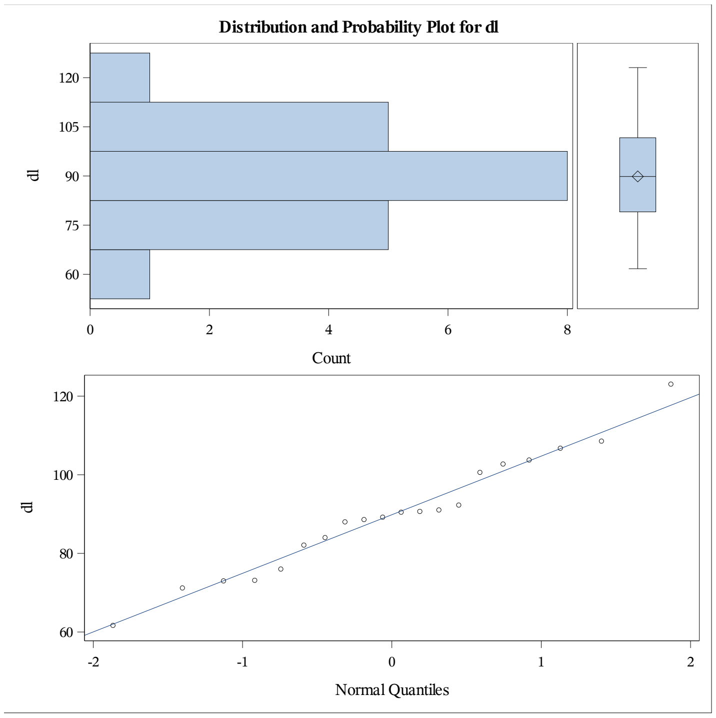
Example. Researchers have shown that cigarette smoking has a deleterious effect on lung function. In their study of the effect of cigarette smoking on the carbon monoxide diffusing capacity (DL) of the lung, Ronald Knudson, W. Kaltenborn and B. Burrows found that current smokers had DL readings significantly lower than either ex-smokers or nonsmokers. The carbon monoxide diffusing capacity for a random sample of current smokers was as follows:

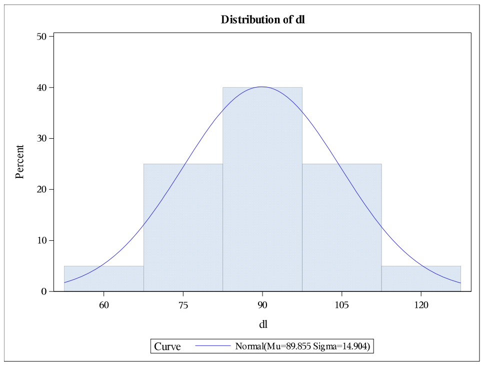
103.768	88.602	73.003	123.086	91.052
92.295	61.675	90.677	84.023	76.014
100.615	88.017	71.210	82.115	89.222
102.754	108.579	73.154	106.755	90.479

Do these data indicate that the mean DL reading for current smokers is lower than 100, the average DL reading for nonsmokers? Test at the $\alpha = 0.01$ level. What is the p-value of the test?

Solution.

(1) The histogram of the data is unimodal and is approximately symmetric. Thus the data should be from a normal population. It can be further verified by QQ plot where most points are falling on the straight line and there is no systematic pattern.





Now we also check the normality by conducting formal statistical tests by testing

H_0: data are from a normal population versus H_a: data are not from a normal population.

The p-values of the test Shapiro-Wilk, Kolmogorov-Smirnov, Cramer-von Mises and Anderson-Darling are all very large shown in the following table. These large p-values cannot warrant the rejection of H_0: data are from a normal population.

Tests for Normality

Test	Statistic		p Value	
Shapiro-Wilk	W	0.977903	Pr < W	0.9042
Kolmogorov-Smirnov	D	0.134969	Pr > D	>0.1500
Cramer-von Mises	W-Sq	0.045419	Pr > W-Sq	>0.2500
Anderson-Darling	A-Sq	0.254421	Pr > A-Sq	>0.2500

Therefore, we conclude that the data are from a normal population.

(2)

We are testing

$$H_0: \mu=100 \text{ versus } H_a: \mu<100.$$

The test statistic of the t-test is -3.04 with a p-value 0.0033 which of course is smaller than the significance level 0.05. Thus, H_0 is rejected. There is sufficient evidence to show that the mean DL reading for current smokers is lower than 100.

DF	t Value	Pr < t
19	-3.04	0.0033

SAS code

```
data cig;
input dl@@;
cards; /* or use 'datalines;'*/
103.768   88.602   73.003   123.086   91.052
92.295    61.675   90.677   84.023    76.014
100.615   88.017   71.210   82.115    89.222
102.754   108.579   73.154   106.755   90.479
;
run;

proc univariate data=cig normal plot; /* check normality*/
var dl;
histogram dl/normal;
run;

proc ttest sides=L data=cig H0=100;
var dl;
run;
```

Bivariate Normal Distribution

Density Surface and Contour Plot of a Bivariate Normal Distribution

The density surface of the standard bivariate normal distribution (each marginal has mean 0 and standard deviation 1) is produced by the following SAS code. The correlation is specified in the third line of the SAS code (here at 0.6). It would be a good idea to try this program for various values of r between -1 and 1 to explore how the shape of the normal distribution varies with the correlation.

The correlation ρ describes the contour ellipses of this density. If both variables are scaled to have a variance of 1, a correlation of zero corresponds to circular contours, whereas the ellipses become narrower and finally collapse into a line segment as the correlation approaches 1 or -1.

SAS code:

```
options ls=78;

title “Bivariate Normal Density”;

%let r=0.6;

data binormal;

pi=3.1416;

do x=-4 to 4 by 0.1;

do y=-4 to 4 by 0.1;

phi=(1/(2*pi))*(1/sqrt(1-&r*&r))*exp(-(x**2-2*&r*x*y+y**2)/(2*(1-
&r*&r)) );

output;

end;

end;

proc g3d data= binormal;

plot x*y=phi/ rotate=-20;

run;

proc gcontour data=binormal;

plot x*y=phi;

run;
```

Hypothesis Test of Population Linear Correlation ρ

The following SAS code is used to test the population linear correlation ρ

```
data bmi;

input weight height age@@;

cards; /*or use datalines;*/

64 57 8

71 59 10

53 49 6

67 62 11

55 51 8

58 50 7

77 55 10

57 48 9

56 42 10

51 42 6

76 61 12

68 57 9

;

run;


proc corr data=bmi pearson fisher(rho0=0 TYPE=TWOSIDED);

TITLE "Example of a Pearson Correlation";

   var weight height;

run;
```

Fitting and Inference for Simple Linear Regression Models

1. Example of Fitting an SLR. We fit the SLR model for the bmi example.

```
data bmi;
input weight height age@@;
datalines;
64 57 8 71 59 10 53 49 6 67 62 11
55 51 8 58 50 7 77 55 10 57 48 9
56 42 10 51 42 6 76 61 12 68 57 9
;
run;

proc plot data=bmi;
plot weight*height ='*';
run;
```

```
proc reg data=bmi;
TITLE "Regression Line for Height-Weight Data";
   model weight = height;
run;
```

We may like to put the scatter plot and the fitted regression line in the same graph:

```
proc reg data=bmi;
```

```
model weight = height;
ods output ParameterEstimates=PE;
run;

data _null_;
set PE;
if _n_ = 1 then call symput('Int', put(estimate, BEST6.));
else call symput('Slope', put(estimate, BEST6.));
run;

proc sgplot data=bmi noautolegend;
title "Regression Line with Slope and Intercept";
reg y=weight x=height;
inset "Intercept = &Int" "Slope = &Slope" /
border title="Parameter Estimates" position=topleft;
run;
```

2. Example of Inferences in an SLR model. We fit the SLR model for the mileage example.

```
 data mileage;
infile 'mileage.txt’
firstobs=2;
input miles gallons;
run;
```

```
proc reg data=mileage  ALPHA=0.05; /* significance level for confidence and prediction intervals*/

model gallons = miles /clb;  /*CLB computes confidence limits for the parameter estimates; CLM computes confidence limits for the expected value of the dependent variable; CLI computes confidence limits for for an individual predicted value;  */

run;
```

It is possible to let SAS do the predicting of new observations and/orestimating of mean responses. The way to do this is to enter the values of the independent variables you are interested in during the data input step,but put a period (.) for the unknown y value. That is,

```
data newobs;

input miles gallons@@;

datalines;

340 .

;

run;


data mileage;

set mileage newobs;

run;


proc reg data=mileage;

model gallons = miles / r cli clm; /* r produces analysis of residuals */

run;
```

We can use proc iml to remove the new observation from the data set.

```
proc iml;

edit mileage;

delete point 36;

run;

quit;
```

3. Example of Residual Analysis.

```
proc reg simple data=mileage; /* “simple” displays simple statistics*/

model gallons = miles;

output out=new p=predict r=resid;

run;


proc plot data=new;

plot resid*miles ='+ ';

plot resid*predict = '. ';

run;


proc univariate data=new normal plot;

var resid;

histogram resid/normal;

run;


/*add an  influential point to the data set */

data newobs;
```

```
input miles gallons@@;
datalines;
370 4
;
run;

data mileage;
set mileage newobs;
run;

/*fit the regression model again*/
 proc reg simple data=mileage;
model gallons = miles;
run;

/*proc delete can be used to delete a data set*/
proc delete data=newobs;
run;

proc reg data=mileage;
model gallons = miles / DW; /* DW computes a Durbin-Watson statistic */
run;
```

4. Example of Lack of Fit Test.

Suppose the following data are the number of hours a student spent studying for his/her weekly quiz and his/her corresponding exam scores.

0 hours		3 hours		5 hours	6 hours		8 hours		10 hours		
22	29	60	65	60	64	82	76	81	89	92	86

```
data quizzes;
input time score @@;
datalines;
0 22 0 29 3 60 3 65
5 60 6 64 6 82 8 76
8 81 10 89 10 92 10 86
;
run;

proc plot data= quizzes;
plot score*time ='*';
run;

proc reg data= quizzes;
model score = time / lackfit;
run;
```

Simultaneous Inferences for Simple Linear Regression Models

Bonferroni Joint Confidence Intervals.

Example (Exercise 1.22). Sixteen batches of the plastic were made, and from each batch one test item was molded and the hardness measured at some specific point in time. The results are shown in PlasticHardness.txt; X is elapsed time in hours, and Y is hardness in Brinell units. Obtain 90% Bonferroni joint confidence intervals for β_0 and β_1 using the code below:

```
data plastic;
input y x @@;
datalines;
199.0  16.0  205.0  16.0  196.0  16.0
200.0  16.0  218.0  24.0  220.0  24.0
215.0  24.0  223.0  24.0  237.0  32.0
234.0  32.0  235.0  32.0  230.0  32.0
250.0  40.0  248.0  40.0  253.0  40.0
246.0  40.0
;
run;

 proc reg data=plastic;
model y = x;
run;

data intervals;
tvalue = tinv(1-0.025,14);
b0lo=168.60000-(tvalue*2.65702); /*lower bound for b0*/
```

```
b0up=168.60000+(tvalue*2.65702); /*upper bound for b0*/

b1lo=2.03438-(tvalue*0.09039); /*lower bound for b1*/

b1up=2.03438+(tvalue*0.09039); /*upper bound for b1*/

run;


proc print data=intervals;

var b0lo b0up b1lo b1up;

run;


/* noint fits a regression model without intercept */

proc reg data=plastic;

model y = x/noint clb alpha=0.01;

run;
```

Bootstrapping Regression Model

1. One-sample bootstrapping

```
data onesample;
input y@@;
datalines;
6 -3 5 3
;
run;

proc ttest data=onesample H0=3;  /* test H0: mu=3*/
var y;
run;

/* Now create 300 copies of the original data set */
data replicates;
set onesample;
  do sample=1 to 300;
output;
keep sample y;
end;
run;

proc sort data=replicates;
by sample;
run;

proc print data=replicates (obs=40); /*print first 10 samples*/
run;

/*Now sample with replacement */
proc surveyselect data= replicates method=urs sampsize=4 rep=300 outhits
out=bootsamples;
id y;
run;

proc print data=bootsamples (obs=40);
run;

proc sort data= bootsamples;
by Replicate;
run;

proc print data=bootsamples (obs=40);
run;
```

```
/* Compute the statistic for each bootstrap sample */
proc means data=bootsamples noprint;
   by Replicate;
   var y;
   output out=OutStats mean=Mean;
run;

/* the distribution of the bootstrap sample means */
proc sgplot data=OutStats;
      histogram Mean;
run;
/* the distribution of the bootstrap sample means */
proc univariate data=OutStats;
    var Mean;
    histogram Mean;
    output out=intervals pctlpre=perc_ pctlpts=2.5,5,95,97.5;
run;

proc print data=intervals;
run;
```

2. We use the following example to illustrate the procedure of the Fixed-X bootstrapping method.

```
data a1;
input x y @@;
datalines;
39 65 43 78
21 52 64 82
57 92 47 89
28 73 75 98
34 56 52 75
;
run;

/* fit the model*/
proc reg data=a1;
model y=x / clb;
output out=a2 p=pred r=res;
run;

proc print data=a2;
run;
```

```
/* Now create numerous copies of the original data set */
data pred;
set a2;
  do sample=1 to 1000;
output;
keep sample x y pred;
end;
run;

proc sort data=pred;
by sample;
run;

proc print data=pred (obs=100); /*print first 10 samples*/
run;


/*Now sample with replacement from the residuals*/
proc surveyselect data=a2 method=urs sampsize=10 rep=1000 outhits out=res;
id res;
run;


proc print data=res (obs=100); /*print first 10 samples*/
run;


data new;
 merge pred res;
 ynew = pred + res;
run;

proc print data=new (obs=20);
run;
/* Perform regression on each sample data set and store parameter
   estimate results in a dataset called parm.  The ods listing turns
   off the output going into the output window.*/

ods html close;
proc reg data=new;
model ynew=x;
 by sample;
ods output ParameterEstimates=parm;
run;
ods html;
```

```
proc print data=parm (obs=10);
run;

/*delete all intercept estimates using proc IML*/
proc iml;
edit parm;
delete all where(variable='Intercept');
run;
quit;

/* Generate histogram and approximate the density */
proc univariate noprint data=parm;
var Estimate;
histogram Estimate / kernel ;
output out=a4 mean=bmean std=bsterr pctlpre=perc_ pctlpts=2.5,5,95,97.5;
run;

proc print data=a4;
run;
```

Inferences for Multiple Linear Regression Models

Example

Let’s see what happens when we identify and fit a MLR model to data in Cars93a.txt (the complete data set is in cars93.xls). The scatterplot array shows the response, highway mpg (HIGHMPG) and three potential predictors, displacement (DISPLACE), horsepower (HP) and rpm (RPM).

```
data Cars93a;
infile 'F:\Data\cars93a.txt' firstobs=2;
input HIGHMPG DISPLACE HP RPM;
run;
```

```
proc sgscatter data=Cars93a;
title "Scatterplot Matrix for Cars Data";
matrix HIGHMPG DISPLACE HP RPM;
run;
```

There seems to be a curvilinear relation between the response and each potential predictor, so we will begin by trying a model with linear and squared terms for each predictor.

```
 data Cars93new;
set Cars93a;
D=DISPLACE;
D2=DISPLACE**2;
H=HP;
H2=HP**2;
R=RPM;
R2=RPM**2;
run;
proc reg simple data=Cars93new;
model HIGHMPG = D D2 H H2 R R2;
run;
```

To include the confidence intervals (99% confidence interval for example) of the regression parameters, we use the following SAS code.

```
proc reg data=Cars93new;
model HIGHMPG = D D2 H H2 R R2 / clb alpha=0.01;
run;
```

To get confidence intervals of the mean response and the prediction confidence intervals of an individual response (99% confidence interval for example) of the regression parameters, we use the following SAS code.

```
proc reg data=Cars93new;
model HIGHMPG = D D2 H H2 R R2 / clm cli alpha=0.01;
run;
```

To check model assumptions, we consider QQ plot of the residuals and residual plot:

```
proc reg data=Cars93new;
model HIGHMPG = D D2 H H2 R R2;
output out=new p=predict r=resid;
run;

proc univariate normal plot data=new;
var resid;
histogram resid/normal;
run;

proc plot data=new;
plot resid*predict= 'o';
run;
```

Inferences for Multiple Linear Regression Models

Example 1

Table 7.1 in contains data for a study of the relation of body fat (Y) to triceps skinfold thickness (X1), thigh circumference (X2), and midarm circumferences (X3) based on a sample of 20 healthy females 25-34 years old. Note that the triceps skinfold thicknesses X1 and thigh circumferences X2 for these subjects are highly correlated. The following is the data set.

```
data ch7tab01;
input X1 X2 X3 Y@@;
label x1 = 'Triceps'
x2 = 'Thigh cir. '
x3 = 'Midarm cir. '
y = 'body fat';
cards;
19.5 43.1 29.1 11.9
24.7 49.8 28.2 22.8
30.7 51.9 37.0 18.7
29.8 54.3 31.1 20.1
19.1 42.2 30.9 12.9
25.6 53.9 23.7 21.7
31.4 58.5 27.6 27.1
27.9 52.1 30.6 25.4
22.1 49.9 23.2 21.3
25.5 53.5 24.8 19.3
31.1 56.6 30.0 25.4
30.4 56.7 28.3 27.2
18.7 46.5 23.0 11.7
19.7 44.2 28.6 17.8
14.6 42.7 21.3 12.8
29.5 54.4 30.1 23.9
27.7 55.3 25.7 22.6
```

```
30.2 58.6 24.6 25.4
22.7 48.2 27.1 14.8
25.2 51.0 27.5 21.1
;
run;
```

Let's consider the following 4 models:

```
proc reg data = ch7tab01;
model y = x1;
model y = x2;
model y = x1 x2;
model y = x1 x2 x3;
run;
```

X1 and X2 are highly correlated, so one of them should be removed from the model. Furthermore, when we fit the full model, it seems that X3 does not contribute to the model significantly. In the following, Test1 is for the testing one variable, and Test2 is for the testing two variables at once.

```
proc reg data = ch7tab01;
model y = x1 x2 x3/ss1;
test1: test x3 = 0;
test2: test x2=x3=0;
run;
```

In the following, $r^2_{y1|2}$ and $r^2_{y2|1}$ are from the first model. $r^2_{y3|12}$ is coefficient from X_3 in the second model.

```
proc reg data = ch7tab01;
model y = x1 x2 / pcorr2;
model y = x1 x2 x3 / pcorr2;
run;
```

Example 2. Correlation Transformation

```
data correlation1;
input x1 x2@@;
cards;
10 10 10 10 10 15 10 15
15 10 15 10 15 15 15 15
;
run;
proc corr data=correlation1;
var x1 x2;
run;

data correlation2;
input x1 x2@@;
cards;
10.0 10.0 11.0 11.4 11.9 12.2 12.7 12.5
13.3 13.2 14.2 13.9 14.7 14.4 15.0 15.0
;
run;
proc corr data=correlation2;
var x1 x2;
run;

proc iml;
A = { 1 0.99215, 0.99215 1};
print A;
B=inv(A);
print B;
quit;
```

Example 3. Multicollinearity and Its Effects

```
proc sql;   /*correlation transformation; create a data set temp*/
create table temp as
select *, ( y - mean(y) )/( std(y)*( sqrt( count(y)-1 ) ) ) as yprime,
( x1 - mean(x1) )/( std(x1)*( sqrt( count(x1)-1 ) ) ) as x1prime,
( x2 - mean(x2) )/( std(x2)*( sqrt( count(x2)-1 ) ) ) as x2prime,
( x3 - mean(x3) )/( std(x3)*( sqrt( count(x3)-1 ) ) ) as x3prime
from ch7tab01;
quit;


proc means data = temp mean std ;
var y x1 x2 x3;
var yprime x1prime x2prime x3prime;
run;


/* print the transformed data */
proc print data = temp;
var yprime x1prime x2prime x3prime;
run;

proc reg data = temp corr;
model yprime = x1prime x2prime x3prime;
run;


proc reg data = temp corr;
model yprime = x1prime x3prime;
run;
```

Inferences for Multiple Linear Regression Models

Example: Categorical Explanatory Variables

Table 7.1 in contains data for a study of the relation of body fat (Y) to triceps skinfold thickness (X1), thigh circumference (X2), and midarm circumferences (X3) based on a sample of 20 healthy females 25-34 years old. Note that the triceps skinfold thicknesses X1 and thigh circumferences X2 for these subjects are highly correlated. The following is the data set.

```
data ch8tab02;
input y x1 x2@@;
label y = 'Months'
x1 = 'Size'
x2 = 'Firm Indicator';
cards;
17 151 0 26 92 0
21 175 0 30 31 0
22 104 0 0 277 0
12 210 0 19 120 0
4 290 0 16 238 0
28 164 1 15 272 1
11 295 1 38 68 1
31 85 1 21 224 1
20 166 1 13 305 1
30 124 1 14 246 1
;
run;
proc reg data = ch8tab02;
model y = x1 x2/ clb;
run;
```

```
data ch8tab02;

set ch8tab02;

if x2 = 0 then do;

z1 = x1;

y1 = y;

end;

if x2= 1 then do;

z2 = x1;

y2 = y;

end;

run;


proc reg data = ch8tab02;

model y = z1;

run;


proc reg data = ch8tab02;

model y = z2;

run;
```

Building a Multiple Linear Regression Model

Example: Surgical Unit Example

Table 7.1 in contains data for a study of the relation of body fat (Y) to triceps skinfold thickness (X1), thigh circumference (X2), and midarm circumferences (X3) based on a sample of 20 healthy females 25-34 years old. Note that the triceps skinfold thicknesses X1 and thigh circumferences X2 for these subjects are highly correlated. The following is the data set.

```
data ch9tab01;

input x1 x2 x3 x4 y@@;

label x1 = 'blood-clotting'

x2 = 'prognostic'

x3 = 'enzyme'

x4 = 'liver function'

y = 'survival';

cards;

6.7  62   81  2.59  200 5.1  59   66  1.70  101

7.4  57   83  2.16  204 6.5  73   41  2.01  101

7.8  65  115  4.30  509 5.8  38   72  1.42   80

5.7  46   63  1.91   80 3.7  68   81  2.57  127

6.0  67   93  2.50  202 3.7  76   94  2.40  203

6.3  84   83  4.13  329 6.7  51   43  1.86   65

5.8  96  114  3.95  830 5.8  83   88  3.95  330

7.7  62   67  3.40  168 7.4  74   68  2.40  217

6.0  85   28  2.98   87 3.7  51   41  1.55   34

7.3  68   74  3.56  215 5.6  57   87  3.02  172

5.2  52   76  2.85  109 3.4  83   53  1.12  136

6.7  26   68  2.10   70 5.8  67   86  3.40  220

6.3  59  100  2.95  276 5.8  61   73  3.50  144

5.2  52   86  2.45  181 1.2  76   90  5.59  574

5.2  54   56  2.71   72 5.8  76   59  2.58  178

3.2  64   65  0.74   71 8.7  45   23  2.52   58
```

```
5.0  59   73  3.50  116 5.8  72   93  3.30  295
5.4  58   70  2.64  115 5.3  51   99  2.60  184
2.6  74   86  2.05  118 4.3   8  119  2.85  120
4.8  61   76  2.45  151 5.4  52   88  1.81  148
5.2  49   72  1.84   95 3.6  28   99  1.30   75
8.8  86   88  6.40  483 6.5  56   77  2.85  153
3.4  77   93  1.48  191 6.5  40   84  3.00  123
4.5  73  106  3.05  311 4.8  86  101  4.10  398
5.1  67   77  2.86  158 3.9  82  103  4.55  310
6.6  77   46  1.95  124 6.4  85   40  1.21  125
6.4  59   85  2.33  198 8.8  78   72  3.20  313
;
run;

proc sgscatter data=ch9tab01;
title "Scatterplot Matrix for Surgical Data";
matrix y x1 x2 x3 x4;
run;

proc reg data=ch9tab01;
model y = x1 x2 x3 x4;  /*first-order regression model*/
run;

data ch9tab01;
set ch9tab01;
logy = log(y);
run;
```

```
proc sgscatter data=ch9tab01;
title "Scatterplot Matrix for Surgical Data";
matrix logy x1 x2 x3 x4;
run;

proc reg data=ch9tab01 corr;
model logy = x1 x2 x3 x4;  /*first-order regression model*/
run;

/*Stepwise Regression Method*/
proc reg data = ch9tab01;
model logy = x1-x4/ selection = stepwise slentry= .01 slstay= .05;
run;

proc reg data=ch9tab01;
model logy = x1-x4/ selection=rsquare adjrsq cp mse sse aic sbc;
run;  /*PRESSp is not a selection criterion in SAS*/

proc reg data=ch9tab01 OUTEST=est PRESS;
model logy = x1 x2 x3 x4;
run;
proc print data=est;
run;
```

Building a Multiple Linear Regression Model

Example 1: Surgical Unit Example

```
data ch10tab01;
input x1 x2 y@@;
label x1 = 'Income'
x2 = 'Risk Aversion'
y = 'Insurance';
cards;
45.010 6 91 57.204 4 162
26.852 5 11 66.290 7 240
40.964 5 73 72.996 10 311
79.380 1 316 52.766 8 154
55.916 6 164 38.122 4 54
35.840 6 53 75.796 9 326
37.408 5 55 54.376 2 130
46.186 7 112 46.130 4 91
30.366 3 14 39.060 5 63
;
run;
proc reg data = ch10tab01 ;
model y = x1 x2 / partial ;  /*partial regression plots*/
plot r.*x1;
run;
```

The residual plot clearly suggests that a linear relation for X1, is not appropriate in the model already containing X2. To obtain more information about the nature of this relationship, we shall use an added-variable plot. We regress Y and X1, each against X2.

```
proc reg data=ch10tab01 noprint; /* fit without output*/

model y x1 = x2 ;

output out=tempx1 r=ry rx;

run;


proc gplot data=tempx1;

plot ry*rx;

label ry='e(Y|X2) '

rx='e(X1|X2) ';

run;
```

The added-variable plot suggests that the curvilinear relation **between *Y* and *X1*** when *X2* is already in the regression model is strongly positive, and that a slight concave upward shape may be present.

Example 2: Surgical Unit Example

Table 7.1 in contains data for a study of the relation of body fat (Y) to triceps skinfold thickness (X1), thigh circumference (X2), and midarm circumferences (X3) based on a sample of 20 healthy females 25-34 years old. Note that the triceps skinfold thicknesses X1 and thigh circumferences X2 for these subjects are highly correlated. The following is the data set.

```
data ch9tab01;

input x1 x2 x3 x4 y@@;

logy = log(y);

label x1 = 'blood-clotting'
```

```
x2 = 'prognostic'
x3 = 'enzyme'
x4 = 'liver function'
y = 'survival';
logy = 'logSurvival';
cards;
6.7  62   81  2.59  200 5.1  59   66  1.70  101
7.4  57   83  2.16  204 6.5  73   41  2.01  101
7.8  65  115  4.30  509 5.8  38   72  1.42   80
5.7  46   63  1.91   80 3.7  68   81  2.57  127
6.0  67   93  2.50  202 3.7  76   94  2.40  203
6.3  84   83  4.13  329 6.7  51   43  1.86   65
5.8  96  114  3.95  830 5.8  83   88  3.95  330
7.7  62   67  3.40  168 7.4  74   68  2.40  217
6.0  85   28  2.98   87 3.7  51   41  1.55   34
7.3  68   74  3.56  215 5.6  57   87  3.02  172
5.2  52   76  2.85  109 3.4  83   53  1.12  136
6.7  26   68  2.10   70 5.8  67   86  3.40  220
6.3  59  100  2.95  276 5.8  61   73  3.50  144
5.2  52   86  2.45  181 1.2  76   90  5.59  574
5.2  54   56  2.71   72 5.8  76   59  2.58  178
3.2  64   65  0.74   71 8.7  45   23  2.52   58
5.0  59   73  3.50  116 5.8  72   93  3.30  295
5.4  58   70  2.64  115 5.3  51   99  2.60  184
2.6  74   86  2.05  118 4.3   8  119  2.85  120
4.8  61   76  2.45  151 5.4  52   88  1.81  148
```

```
5.2  49   72  1.84   95 3.6  28   99  1.30   75
8.8  86   88  6.40  483 6.5  56   77  2.85  153
3.4  77   93  1.48  191 6.5  40   84  3.00  123
4.5  73  106  3.05  311 4.8  86  101  4.10  398
5.1  67   77  2.86  158 3.9  82  103  4.55  310
6.6  77   46  1.95  124 6.4  85   40  1.21  125
6.4  59   85  2.33  198 8.8  78   72  3.20  313
;
run;

proc reg data = ch9tab01;
model y = x1-x4/ influence;
output out=RegOut dffits=DFFits cookd=CooksD;
run;

proc print data=RegOut;
run;

proc reg data = ch9tab01;
model y = x1-x4/ vif tol stb collin;
run;

proc corr data=ch9tab01;
var x1 x2 x3 x4;
run;
```

Building a Multiple Linear Regression Model

Example 1: Weighted Least Squares

```
data ch11tab01;

input x y@@;

label x='age'

y='Dbp';

cards;

27 73 21 66 22 63 24 75

25 71 23 70 20 65 20 70

29 79 24 72 25 68 28 67

26 79 38 91 32 76 33 69

31 66 34 73 37 78 38 87

33 76 35 79 30 73 31 80

37 68 39 75 46 89 49 101

40 70 42 72 43 80 46 83

43 75 44 71 46 80 47 96

45 92 49 80 48 70 40 90

42 85 55 76 54 71 57 99

52 86 53 79 56 92 52 85

50 71 59 90 50 91 52 100

58 80 57 109

;

run;
```

```
proc reg data=ch11tab01;

model y = x;

output out=temp r=resid;

plot y*x r.*x;

run;
```

The nonconstant error variance can be seen from the residual plot. Now let's see the absolute residuals plot.

```
data temp;

set temp;

absr = abs(resid);

run;

proc gplot data = temp;

plot absr*x;

run;
```

```
/*gplot is not supported by SAS University Edition*/
proc sgplot data = temp;
scatter x=x y=absr;
run;
```

It seems that there is a linear relation between the error standard deviation and X. We therefore regressed the absolute residuals against X and record the predicted values which are the estimated expected standard deviation.

```
proc reg data = temp ;

model absr = x;
```

```
output out = temp1 p = s;

run;
```

The fitted model is

$$\hat{s} = -1.5495 + 0.19817x.$$

The weights are then obtained by using

$$w_i = \frac{1}{(\hat{s}_i)^2}$$

```
data temp1;

set temp1;

w = 1/(s**2);

run;


proc print data = temp1 (obs = 10);

run;
```

Last, we fit the regression model using weighted least squares.

```
proc reg data = temp1;

weight w;

model y = x / clb;

run;
```

It is interesting to note that the standard deviation is somewhat smaller than the standard deviation of the estimate obtained by ordinary least squares method.

Example 2: Ridge Regression

```
data ch7tab01;

input X1 X2 X3 Y;

label x1 = 'Triceps'

x2 = 'Thigh cir. '

x3 = 'Midarm cir. '

y = 'body fat';

cards;

19.5 43.1 29.1 11.9

24.7 49.8 28.2 22.8

30.7 51.9 37.0 18.7

29.8 54.3 31.1 20.1

19.1 42.2 30.9 12.9

25.6 53.9 23.7 21.7

31.4 58.5 27.6 27.1

27.9 52.1 30.6 25.4

22.1 49.9 23.2 21.3

25.5 53.5 24.8 19.3

31.1 56.6 30.0 25.4

30.4 56.7 28.3 27.2

18.7 46.5 23.0 11.7

19.7 44.2 28.6 17.8

14.6 42.7 21.3 12.8

29.5 54.4 30.1 23.9

27.7 55.3 25.7 22.6

30.2 58.6 24.6 25.4
```

```
22.7 48.2 27.1 14.8

25.2 51.0 27.5 21.1

;

run;


/*correlation transformation of data*/

proc sql; create table ch7tab1a as

select *, ( y - mean(y) )/( std(y)*( sqrt( count(y)-1 ) ) ) as ty,

( x1 - mean(x1) )/( std(x1)*( sqrt( count(x1)-1 ) ) ) as tx1,

( x2 - mean(x2) )/( std(x2)*( sqrt( count(x2)-1 ) ) ) as tx2,

( x3 - mean(x3) )/( std(x3)*( sqrt( count(x3)-1 ) ) ) as tx3

from ch7tab01;

quit;


/*Ridge Regression on Body fat data*/

ods graphics on;

proc reg data = ch7tab1a outest = temp outstb outvif;

model y = x1 x2 x3/ ridge = (0.001 to 0.1 by .01);

run;



proc print data=temp;

run;
```

Here, the outstb option in the proc statement tells SAS to put the parameter estimates in the output temp. These can then be chosen by specifying RIDGESTB in the where statement of the proc print.

```
proc print data = temp;

where _type_ = 'RIDGESTB';

var _ridge_ x1 x2 x3;

run;
```

The outvif option in the proc statement of the regression tells SAS to put the **VIF**'s in the output temp. These can then be chosen by specifying RIDGEVIF in the where statement of the proc print.

```
proc print data = temp;

where _type_ = 'RIDGEVIF';

var _ridge_ x1 x2 x3;

run;
```

Example 3: Robust Regression

```
data ch11tab4;

input state$ y x1 x2 x3 x4 x5;

label y = 'Math profeciency'

        x1 = 'Parents'

        x2 = 'Homelib'

        x3 = 'Reading'

        x4 = 'TV Watching'

        x5 = 'Absences';
```

```
cards;
Alabama 252 75 78 34 18 18
Arizona 259 75 73 41 12 26
Arkansas 256 77 77 28 20 23
California 256 78 68 42 11 28
Colorado 267 78 85 38 9 25
Connecticut 270 79 86 43 12 22
Delaware 261 75 83 32 18 28
Florida 255 75 73 31 19 27
Georgia 258 73 80 36 17 22
Guam 231 81 64 32 20 28
Hawaii 251 78 69 36 23 26
Idaho 272 84 84 48 7 21
Illinois 260 78 82 43 14 21
Indiana 267 81 84 37 11 23
Iowa 278 83 88 43 8 20
Kentucky 256 79 78 36 14 23
Louisiana 246 73 76 36 19 27
Maryland 260 75 83 34 19 27
Michigan 264 77 84 31 14 25
Minnesota 276 83 88 36 7 20
Montana 280 83 88 44 6 21
Nebraska 276 85 88 42 9 19
Ohio 264 79 84 36 11 22
Oklahoma 263 78 78 37 14 22
Oregon 271 81 82 41 9 31
```

```
Pennsylvania 266 80 86 34 10 24

Texas 258 77 70 34 15 18

Virginia 264 78 82 33 16 24

Wisconsin 274 81 86 38 8 21

Wyoming 272 85 86 43 7 23

;

run;
```

PROC ROBUSTREG provides four estimation methods: M estimation, LTS estimation, S estimation, and MM estimation. The default method is M estimation (IRLS Robust Regression).

```
proc reg data = ch11tab4;

model y = x2;

plot y*x2 r.*x2;

run;


proc robustreg data=ch11tab4 method=m (wf=huber);

model y = x2;

output out=new p=predict r=resid;

run;


proc plot data=new;

plot resid*x2 ='+';

run;
```