AN INTRODUCTION TO SAS

The SAS (Statistical Analysis System) System is a software system for data analysis and report writing. SAS is one of the most widely used statistical packages in the world. In addition to statistics, The SAS System is an integrated system of software providing complete control over data management, analysis, and presentation. Although originally designed for mainframe computer, several platform versions are now available, including PC (MS-DOS, Windows), OS/2, and Unix.

Base SAS software is the foundation of the SAS System. Base SAS includes SAS language, basic SAS procedure, SAS Macro language, SAS Display Manager System, and SAS Text Editor. With base SAS software you can store data values and retrieve them, modify data, compute simple statistics, and create reports all in one SAS session. The other modules that are interested to general biologists are STAT (statistics), GRAPH (graphic display), and IML (matrix language), which provides graphics, forecasting, data entry, and sophisticated statistics.

Why use SAS?

SAS has several important advantages over other statistical software designs. SAS is a programming language, many syntax are similar to QuickBASIC. You can write your own program and run it many times for different data sets. SAS is very complete in terms of statistical methods. SAS is extremely powerful in sorting, arranging, and manipulating raw data. Two or more data sets recorded in different formats can be easily merged together for analysis. Data can be read from and transferred to files resides in hard disks or floppies, Lotus 1-2-3, dBASE files, or input from keyboard.

The disadvantages of using SAS include: SAS is not user friendly. It takes time to learn SAS. You'll need to know how to write a computer program. The SAS' documents are lengthy, and poorly organized. Information is sometimes scattered over several chapters or volumes. PC SAS version (6.04) takes several megabytes of hard disks space to store and to get better performance, you'll need a 386 or 486 machine.

Running SAS in PC

A typical SAS program may look like this one that does analysis of variance:

```sas
/* ** This is a comment line **
 ** this is also a comment line **
 ** this indicate end of comment ** */

data Squirrel;
  infile 'Squirrel.DAT';
  input Sex $ Age Weight;
run;

proc Sort;
  by Sex;
run;

* this is another type of comment line;
proc GLM;
  class Sex;
  model Weight = Sex;
run;
```
In general, SAS programs take this form. There are two steps in SAS. The first is you create a data set, read and sort the data, and make the data format compatible to the procedure (PROC). The second step is then that the procedure analyzes and prints the results. An important thing to remember is that all SAS statements must end with a semi-colon (;).

**Basic Commands for a SAS Program**

A sequence of command is necessary to indicate the file name and data to be read in using SAS.

**Getting data into SAS data set**

Data is set up in SAS by the following sequence of commands:

```
the DATA statement
the INPUT statement
the INFILE statement
the RUN statement
```

The first command is `DATA filename;` where the file name is the name the user gives to identify the data set in SAS. Examples are `DATA Ecology;` or `DATA Behavior;`. Data set names must begin with a letter (A-Z, a-z) and be 8 characters or less.

The next command is the input statement, where `INPUT Var1 Var2 ...;` is the free format reading of the data. For character data use a $ and indicate column locations of the variables. For example

```
INPUT Name $ 1-5;
```

indicates that name is a character variable located in column 1-5. If the variable is numeric the name and column location are given. For example

```
INPUT Id 1-3;
```

indicates that the numeric variable ID is located in columns 1-3.

The input command can be continued on the next line but variable names cannot be split. For multiple records per case, use # card number to indicate a new record. For example

```
INPUT Id 1-3 Name 4 5-10 #2 Id 1-3;
```

indicates that the first card has Id and Name while the second card has Id on it.

The input command has a simpler form if the input data satisfies the following criteria

- each of the values on the data line is separated from the next value by at least one blank column.
- character variables have 8 characters or fewer.
- data don't include missing values.
- numeric values include any necessary decimal points.

In this case one specifies `INPUT` followed by the variable name. For example

```
INPUT Id Names SBP DBP Chol;
```
The last is the *INFILE filename* command. The first statements of the program are then

```
DATA filename;
INFILE filename;
INPUT Variables;
```

### Some SAS Procedures

The SAS procedures include

#### Base System

**APPEND** - appends data from one SAS data set to the end of another SAS data set

**CALANDER** - displays data from a SAS data set in a month-by-month calendar format

**CHART** - produces vertical and horizontal bar charts (histograms), block charts, pie charts, and star charts

**COMPARE** - compares contents of two SAS data sets

**CONTENTS** - describes the contents of a SAS data library or specified members of the library

**COPY** - copies a SAS data library or selected members of the library

**CORR** - computes correlation coefficients between variables, including Pearson product-moment and weighted product-moment correlations

**DATASETS** - manages a SAS data library

**DBF** - converts dBASE II, III, and IV files to SAS data sets

**DIF** - converts VisiCalc and Lotus 1-2-3 files to SAS data sets

**DOWNLOAD** - moves SAS data sets from a mainframe computer to a microcomputer

**FORMAT** - defines output formats for value labelling

**FORMS** - produces labels for envelopes, mailing labels, external tape labels, file cards, and any other printer forms that have a regular pattern

**FREQ** - produces one-way to n-way frequency and cross-tabulation tables

**MEANS** - produces simple univariate descriptive statistics for numeric variables

**PLOT** - graphs one variable against another, producing a printer plot

**PRINT** - prints the observations in a SAS data set, using all or some of the variables

**RANK** - computes ranks for one or more numeric variables across the observations of a SAS data set
SORT - sorts a SAS data set according to one or more variables

STANDARD - standardizes some or all of the variables in a SAS data set to a given mean and standard deviation and produces a new SAS data set to contain the standardized values

SUMMARY - computes descriptive statistics on numeric variables in a SAS data set and outputs the results to a new SAS data set

TABULATE - constructs tables of descriptive statistics from compositions of classification variables, analysis variables, and statistics keywords. Tables can have up to three dimensions: column, row, and page

TIMEPLOT - is used to plot one or more variables over time intervals

TRANSPOSE - turns a SAS data set on its side, changing variables into observations and observations into variables

UNIVARIATE - produces simple descriptive statistics (including quantiles) for numeric variables

UPLOAD - moves SAS data sets from a microcomputer to a mainframe computer

Regression

CALIS - fits systems of linear structure equations and path analysis.

CATMOD - analyzes data that can be represented by a contingency table

GLM - uses the method of least squares to fit general linear models

LIFEREG - fits parametric models to failure-time data that may be right-censored

LOGISTIC - fits logistic models

NLIN - build nonlinear regression models

ORTHOREG - performs regression using the Gentleman-Givens computational method

PROBIT - performs probit regression as well as logistic regression and ordinal logistic regression

REG - performs linear regression with many diagnostic capabilities, selects models using one of nine methods, produces scatterplot of raw data and statistics, highlights scatterplot to identify particular observations, and allows interactive changes in both the regression model and the data used to fit the model

RSREG - builds quadratic response-surface regression models

TRANSREG - obtains optimal linear and nonlinear transformations of variables using alternating least squares

Analysis of Variance

ANOVA - handle analysis of variance, multivariate analysis of variance, and repeated measures analysis of variance for balanced designs
CATMOD - fits linear models to functions of categorical data and performs analysis of variance and repeated measures analysis of variance for categorical data

GLM - perform analysis of variance, regression, analysis of covariance, repeated measures analysis, and multivariate analysis of variance

NESTED - performs nested analysis of variance and analysis of covariance for purely nested random models

NPARIWAY - performs nonparametric one-way analysis of rank scores

PLAN - constructs designs and randomizes plans for nested and crossed experiments

TTEST - compares the means of two groups of observations

VARCOMP - estimates variance components for random or mixed models

**Categorical Data Analysis**

CATMOD - fit linear models to functions of categorical data, facilitating such analyses as regression, analysis of variance, linear modeling, log-linear modeling, logistic regression, and repeated measures analysis

CORRESP - performs simple and multiple correspondence analyses, using a contingency table as input or using the raw categorical data as input

FREQ - builds frequency tables or contingency tables and produces a number of tests and measures association such as chi-square statistics, odds ratio, correlation statistics, and Fisher's exact test for any size two-way table

LOGISTIC - fit linear logistic regression models for binary or ordinal response data with maximum-likelihood methods

PROBIT - computes maximum likelihood estimates of regression parameters for binary or ordinal response data

**Multivariate Statistics**

CANCORR - performs a canonical correlation analysis and outputs canonical variable scores

CORRESP - performs correspondence analysis, which is a weighted principal component analysis of a contingency table.

FACTOR - performs principal component and common factor analysis with rotation, and outputs component scores or estimates of common factor scores.

PRINCOMP - performs a principal component analysis and outputs standardized or unstandardized principal component scores.

PRINQUAL - performs a principal components analysis of qualitative data.

TRANSREG - obtains optimal linear and nonlinear transformations of variables using alternating least squares.

Many other SAS/STAT procedures can also analyze multivariate data, for example, CATMOD, GLM, REG, CALIS, TRANSREG, and procedures for cluster and discriminant analysis.
**Discriminant Function Analysis**

CANDISC - performs a canonical analysis to find linear combinations of the quantitative variables that best summarize the differences among the classes.

DISCRIM - computes various discriminant functions for classifying observations.

STEPDISC - uses forward selection, backward elimination, or stepwise selection to try to find a subset of quantitative variables that best reveals differences among the classes.

**Clustering Analysis**

ACECLUS - attempts to estimate the pooled within-cluster covariance matrix from coordinate data without knowledge of the number or the membership of the clusters.

CLUSTER - does hierarchical clustering of observations using eleven agglomerative methods applied to coordinate data or distance data.

FASTCLUS - finds disjoint clusters of observations using a k-means method applied to coordinate data.

MODECLUS - finds disjoint clusters of observations with coordinate or distance data using nonparametric density estimation.

TREE - draws tree diagrams, also called dendrograms or phenograms, using output from the CLUSTER or VARCLUS procedures.

VARCLUS - is for both hierarchical and disjoint clustering of variables by oblique multiple-group component analysis.

**Scoring Analysis**

RANK - ranks the observations of each numeric variable from low to high and outputs ranks or rank scores.

SCORE - constructs new variables that are linear combinations of old variables according to a scoring data set.

STANDARD - standardizes variables to a given mean and standard deviation.

TRANSREG - optimally scales variables so that you can then input these variables to other SAS procedures for further analysis.

**Survival Analysis**

LIFEREG - fits parametric accelerated failure time models to event-time data that may be censored.

LIFETEST - computes nonparametric estimates of the survival distribution and rank tests for the association of the event-time variable with other variables.

**Examples**

Data merging, sorting, and output
options ls=78 ps=54;
libname path 'c:\sas\reprod';

* Create a data file for reproductive analysis by merging two SAS files RMALE.SSD and RFEMALE.SSD;

data reprod;
  set path.rmale path.rfemale;

  * Status represents the reproductive stage:
    Status = 1 when the squirrel is subadult
    = 2 when the squirrel is sexually mature;
  
    if sex = 'f' and RepStage = 2 then Status = 2;
    else if sex = 'm' and RepStage > 2 then Status = 2;
    else Status = 1;

  keep YrMonth sex BodyWt HeadBody Tail Ear Stylform RepStage;
  if Status = 2;
  run;

title 'Descriptive Statistics for Flying Squirrel: Adult';
proc contents position;
  run;

proc print;
  run;

proc print;
  run;

Data transferring

options ls=82 ps=54 nocenter;
libname path 'c:\sas\reprod';
filename dbFile 'C:REPROD2.DBF';

* Data file from reproduction study;
data rmale;
  infile 'c:\sas\rmale.dat';
  input YrMonth sex $ BodyWt Body Tail HindFoot Ear Stylform ExtTsTs TstsLong TstsWide TstsWt EpidWt DiamST RepStage Sperm;
  * for adult only;
  if RepStage > 2;
  if BodyWt = 0 then BodyWt=.;
  if Body = 0 then Body=.;
  if Tail = 0 then Tail=.;
  if HindFoot = 0 then HindFoot=.;
  if Ear = 0 then Ear=.;
  if Stylform = 0 then Stylform=.;
  * unit adjustment;
  TstsWt = TstsWt / 100;
  EpidWt = EpidWt / 100;
  DiamSt = DiamST * 10;
run;

proc means noprint;
  var BodyWt Body TstsWt EpidWt DiamST;
  by YrMonth;
  output out = tmp
    mean = BodyWt Body TstsWt EpidWt DiamST
    stderr = BodyWtSE BodySE TstsWtSE EpidWtSE DiamSTSE;
run;

data final;
  set tmp;
  if YrMonth=8112 then Month = "Dec.";
  if YrMonth=8201 then Month = "Jan.";
  if YrMonth=8202 then Month = "Feb.";
  if YrMonth=8203 then Month = "Mar.";
  if YrMonth=8204 then Month = "Apr.";
  if YrMonth=8205 then Month = "May";
  if YrMonth=8206 then Month = "Jun.";
  if YrMonth=8207 then Month = "Jul.";
  if YrMonth=8208 then Month = "Aug.";
  if YrMonth=8209 then Month = "Sep.";
  if YrMonth=8210 then Month = "Oct.";
  if YrMonth=8211 then Month = "Nov.";
run;

proc dbf db4=dbFile data=fin;
  format BodyWt Body TstsWt EpidWt DiamST
   BodyWtSE BodySE TstsWtSE EpidWtSE DiamSTSE 10.3;
run;
Calculate Species Richness

/** Richness.SAS
  ** Species Richness
  ** input: BI_DAT1
  ** output: BI_DAT9A
  ***/

libname path 'c:\sas\coast';
options ls=75 ps=54 obs=max nocenter;

%let InFile =BI_DAT1;
%let OutFile=BI_DAT9A;
%let Animal =Bird;
%let Measure=Species Richness;
%let Level =60 m;
%let Spec=AMCR AMGO AMRO BEWR BHGR BLGR BRCR BTPI CBCH CEWA;

data path.&OutFile;
  set path.&InFile;
  array Species &Spec;
  do over Species;
    if Species > 0 then Species = 1;
    else Species = 0;
  end;
  Richness = sum (of &Spec);
  drop &Spec;
run;

title "&Animal: Category I - &Measure, &Level Level";
proc contents position;
run;
proc print;
  format Site $2. Trans 1. Dist 4. Trtment $7. Richness 4.;
  var Site Trans Dist Trtment Richness;
run;

Calculate Shannon Index

/** Shannon.SAS
  ** Shannon Index
  ** input: BI_DAT1
  ** output: BI_DAT9B
  ***/

libname path 'c:\sas\coast';
options ls=75 ps=54 obs=max nocenter;

%let InFile =BI_DAT1;
%let OutFile=BI_DAT9B;
%let Animal =Bird;
%let Measure=Shannon Index;
%let Level =60 m;
%let Spec=AMCR AMGO AMRO BEWR BHGR BLGR BRCR BTPI CBCH CEWA;
* Shannon Index per Dist;
data path.&OutFile;
  set path.&InFile;
  Total = sum(of &Spec);
  Array Species &Spec;
  do over Species;
    if Species > 0 then
      Species = Species/Total * log(Species/Total);
    else if Species = 0 then
      Species = 0;
  end;
  Shannon = - Sum (of &Spec);
  Drop &Spec;
run;

title "&Animal: Category I - &Measure, &Level Level";
proc contents position;
run;
proc print;
  format Site $2. Trans 1. Dist 4. Trtmnt $7. Shannon 7.4;
  var Site Trans Dist Trtmnt Shannon;
run;

Calculate Simpson Index

/** Simpson.SAS
 ** Simpson Index
 ** input: BI_DAT1
 ** output: BI_DAT9C
 **/

libname path 'c:\sas\coast';
options ls=75 ps=54 obs=max nocenter;

%let InFile =BI_DAT1;
%let OutFile=BI_DAT9C;
%let Animal =Bird;
%let Measure=Simpson Index;
%let Level  =60 m;
%let Spec=AMCR AMGO AMRO BEWR BHGR BLGR BRCR BTPI CBCH CEWA
  COBU COHA CORA DEJU EVGR GCKI GRJA HAFL HAWO HEWA
  HOWR HUVI MGWA MOQU NOFL OCWA OSFL PISI PIWO PYOW
  PUFI RBNU RECR RSTO RTHA RUHU SOSP STJA SWTH TOWA
  TRSW VATH VGSW WCSP WEBL WEFL WETA WIFI WIWA WIWR
  WREN WNPW;

%Macro Sort;
proc sort;
  by Site Trans Dist Trtmnt;
run;
%Mend Sort;

data &InFile;
  set path.&InFile;
  if Stat <= 0 then do;
    if Site="BE" or Site="GC" or Site="PR" then Trtmnt="CCut/CC";
  end;
run;
if Site="BO" or Site="DR" or Site="WC" then Trtment="Plan/PL";
end;
else if Stat > 0 then do;
    if Site="BE" or Site="GC" or Site="PR" then Trtment="Fore/CC"
    if Site="BO" or Site="DR" or Site="WC" then Trtment="Fore/PL"
end;
if Stat >= -260 and Stat <= -240 then Dist = -250;
else if Stat > -240 and Stat <= -180 then Dist = -200;
else if Stat > -180 and Stat <= -120 then Dist = -140;
else if Stat > -120 and Stat <= -60 then Dist = -80;
else if Stat > -60 and Stat <= 0 then Dist = -20;
else if Stat > 0 and Stat <= 60 then Dist = 40;
else if Stat > 60 and Stat <= 120 then Dist = 100;
else if Stat > 120 and Stat <= 180 then Dist = 160;
else if Stat > 180 and Stat <= 240 then Dist = 220;
else if Stat > 240 and Stat <= 300 then Dist = 280;
else if Stat > 300 and Stat <= 360 then Dist = 340;
else if Stat > 360 and Stat <= 400 then Dist = 390;
keep Site Trtment Trans Dist &Spec;
run;
proc sort data=&InFile out=T1;
    by Site Trans Dist Trtment;
run;
proc means noprint;
    by Site Trans Dist Trtment;
    var &Spec;
    output out=t2 sum=&Spec;
run;
*proc print;
* run;
* Simpson Index per Dist;
data &OutFile;
    set T2;
    Total = sum(of &Spec);
    Array Species &Spec;
    do over Species;
        Species = Species * (Species - 1);
    end;
    if Total <= 1 then Simpson = 0;
    else if Total > 1 then
        Simpson = Sum (of &Spec) / (Total * (Total - 1));
    Drop &Spec _TYPE_;
run;
%Sort;
data path.&OutFile;
    set &OutFile;
run;
title "&Animal: Category I - &Measure, &Level Level";
proc contents position;
run;
proc print;
   format Site $2. Trtment $7. Trans 1. Dist 4. Simpson 7.4;
   var Site Trans Dist Trtment Simpson;
run;