Basic SAS Commands

Daren B. H. Cline
Statistics Department
Texas A&M University

26 May 1999

Table of Contents

1. Introduction

For more information, refer to the SAS Introductory Guide or the SAS/STAT Users Guide, vol. 1 and 2 (© SAS Institute Inc., Cary NC). (See also the SAS Practical User's Guide, © Computing Services Center.)

This guide provides description and usage for the SAS command language as it pertains to inputting and analyzing data. Commands are generally submitted through the PROGRAM EDITOR or with batch files. The interactive approach (INSIGHT) can duplicate most of the basic analyses available in the command language, but certainly not all. Thus, in practice one will use both the command language and INSIGHT. At present the guide discusses INSIGHT only briefly (in Section 6).

GETTING STARTED: University microcomputers which are publicly available (in the Open Access Labs) already have SAS installed. You may install SAS on a computer in your department if your department obtains a license for each installation. Or you may obtain free software and license to install it on your personally owned computer, as long as you are a student. To obtain the license, go to the Software Evaluation & Loan Library (SELL) in 113 Teague Building (862-4104).

Pay close attention to the installation instructions and to the hard disk requirements. For most statistical analyses, including those covered here, the following modules are adequate: CORE, BASE, STAT, INSIGHT and HELP. These modules currently require at least 50MB of hard disk space.

After opening SAS, it initially provides three windows called PROGRAM EDITOR, LOG and OUTPUT. Each of these may be reached at any time from the Globals menu. Discussion of their use follow.

PROGRAM EDITOR: Program Editor Window.
Commands can be entered one or more at a time or read in from a file. Use the File menu to open a source file (*file.sas*) or to save it. (The Tool Bar also has buttons for these operations.) It is always wise to save your commands in a file for future reference.

Once a set of commands has been entered, edited and saved, they may be submitted from the Locals menu or by clicking on the Submit button (or on F8). Make sure there is a `run` statement at the end of the submitted text to ensure SAS executes everything. There are two possibilities. If any text is highlighted then just that text is submitted and the PROGRAM EDITOR remains unchanged. However, if no text is highlighted then all the text is submitted and is cleared from the PROGRAM EDITOR. To retrieve text which has been cleared (for example, to correct a mistake and resubmit it) use Recall Text from the Locals menu (or F4).

When SAS processes the submitted text it puts messages into the LOG window and data analysis results into the OUTPUT window.

LOG: Log Window.
This window contains messages describing the processing of the commands. The messages include error messages, indicate whether data was input or output correctly and provide the dataset names. It should be frequently reviewed to ensure proper analyses.

OUTPUT: Output Window.
All the analyses are provided in this window. Note that the results cannot be edited in this window. For this reason you may wish to save it (to *file.lst*) and edit it in an editor of your choice. Unwanted output may first be cleared by choosing Clear Text in the Edit menu (or Ctrl-E).

HELP: Help Menu.
The Help menu contains all the basic information for executing SAS commands. Select Help-SAS System Main Menu. Information on the SAS DATA Step, for example, is found under SAS Language. All the basic procedures for analyzing data are found under Modeling and Analysis Tools.

The rest of this section provides a few basic preliminaries about the SAS language .

PROCEDURES: SAS uses a number of procedures (identified with `proc`) to analyze data. In addition, the datastep (which begin with `data`) is used to input and manipulate data. The datastep and the procedures are discussed below in Sections 2 and 3, respectively.

VARIABLE and DATASET NAMES: The names of variables and of SAS datasets must consist of 8 or fewer alphanumeric characters; underscored blanks (_) also are allowed. SAS does not distinguish between lower and upper case letters except in the value of a character variable, a label or a title. In the examples below, SAS commands will be given in lower case (e.g., `proc reg`). The names of variables and datasets will be in upper case (e.g., `X`, `DATASET1`).

SEPARATING COMMANDS: Every command ends with a semicolon (`;`). First-time users often find that a missing or misplaced semicolon is a cause of many errors. SAS treats multiple spaces and blank lines as a single space so you may space commands to suit you. Also, you may put more than one command (with semicolons) on the same line.

COMMENTS: If an asterisk (`*`) follows a semicolon (`;`) then everything between the asterisk and the next semicolon is treated as commentary and is not interpreted by SAS. Comments will be used to explain commands in the examples below.

2. The Datastep

DATA STATEMENT: A datastep is used to input, create, modify or combine SAS datasets. Since most SAS procedures require a SAS dataset, this is usually the first step other than specifying options or titles. Datasteps may also appear later in the commands. A datastep usually contains three primary commands: `data` (to name the dataset and signal SAS that a datastep begins), `input` (to input the values of variables from the user's dataset) and `output` (to output an observation to the SAS dataset). The `output` statement is often implicit and therefore optional. Between the `input` and `output` commands, new variables can be created using ordinary mathematical expressions. Looping can also be used. Other commands (`set` and `merge`) can be used for modifying existing SAS datasets, but they will not be discussed here.

SAS DATASETS: SAS datasets are created using SAS. They consist of a sequence of observations. Each observation contains the values of one or more variables named by the user. The internal variable `_N_` is the observation number, starting with `_N_` = 1 for the first observation.

USER'S DATASET: In the simplest case, the user inputs a dataset in which each line corresponds to one observation of the SAS dataset and contains all the information required to create that observation. More general scenarios are possible, however, as the examples below illustrate. The user's dataset can follow the datastep or it can be in a separate file. In the former case, the command `cards` must follow the datastep and the data must begin on the line immediately after the `cards` command. If the data are in a separate file, say *file.dat*, then an `infile` command is required. For example, use

```
data DATASET;
infile 'file.dat';                * Input data file;
```

INPUT STATEMENT: The `input` command is used to input data. It is usually easiest to input in free format, one observation per line. However, other approaches are often useful. See the examples below. Note that a `$` must follow a character variable name.

EXPRESSIONS in the DATASTEP: Many ordinary mathematical expressions can be used to compute new quantities. Logical expressions are indicated with parentheses (e.g., `(X>=25)`) and can be treated as ordinary numbers with values 0 (false) and 1 (true). Also, conditional (`if`) statements and looping (`do`) statements may be used. The examples below illustrate some uses of expressions. Section 5 lists common operators and functions.

EXAMPLE DATASTEPS: These examples are just a small selection of the capabilities of the datastep in SAS. Indeed, some users use SAS solely for data manipulation. The first example is of the simplest form where each line of the user's dataset corresponds to one observation of the SAS dataset. SAS executes the `input` command once for each observation and in such a case the `output` command is implicit and not needed.

```
data DATASET1;                      * Create a dataset named DATASET1;
input NAME $15. SEX $ WT HT;        * NAME & SEX have character values;
                                    * NAME must have 15 characters;
cards;                              * The data begins on the next line;
J. Adams       M  151 69
M. B. Finster  F  104 63
K. C. Lodge    M  208 73
M. J. Newsome  M  185 69
Q. Prevoli     F  132 66
;;;                                 * Indicate end of data;
```

The semicolons at the end of the data are optional, but are useful both as visual markers and to ensure SAS does not misinterpret any lines to follow.

The next example illustrates a formatted input where the columns for each variable are identified. This is especially useful if the user's dataset contains more variables per observation than are needed, but it requires the dataset to be strictly formatted. In this example, the variables are input from a data file and are given labels. The `if` statement is used to select only the observations where the patient is male.

```
data DATASET2;                      * Create a dataset named DATASET2;
infile 'patients.dat';              * Input data file;
input NAME $ 1-15 SEX $ 20 WT 30-35 HT 60-65;
if SEX='M';                         * Keep only male patients;
label NAME='Name of Patient'
   WT='Weight at Admission'
   HT='Height at Admission';        * Label variables for plots;
```

Our third example shows how more than one observation can be read from the same line of data. We also show the calculation of new variables.

```
data DATASET3;                      * Create a dataset named DATASET3;
input WIDTH HEIGHT LENGTH @@;       * Continue reading on same line;
VOL=WIDTH*HEIGHT*LENGTH;
LOGVOL=log10(VOL);
LARGE=((VOL>20) or (LENGTH>5));     * LARGE is binary (0 or 1);
cards;                              * The data begins on the next line;
3.1 2.0 4.0   1.1 1.8 1.8   1.5 1.5 2.0
1.2 1.0 3.0   1.2 1.5 5.8   2.5 2.5 3.0
2.4 2.0 2.8   1.2 1.6 1.8   1.0 1.2 2.0
;;;                                 * Indicate end of data (9 obs.);
```

The fourth example uses an array to input multiple observations per line which have common values for some variables. In this case the first two variables (`TEMP` and `HUMIDITY`) are common for five values of the third variable (`STRENGTH`). Missing values are indicated with a `.` (period). Note that the `output` command is explicitly given.

```
data DATASET4;
array X(I) X1-X5;                   * Define array X with length 5;
input TEMP HUMIDITY X1-X5;
do over X;                          * Begin a loop. I=1,5 is implicit;
```

```
   RUN=I; STRENGTH=X;            * Obtain RUN and STRENGTH;
   output;                       * Output one observation each loop;
end;                             * End the loop;
drop I X1-X5;                    * Drop variables not needed;
cards;                           * 30 obs., including 2 missing;
75 70  14.3 15.9 13.8 15.7 15.2
75 80  13.6 15.3 14.8 13.2 14.6
75 90  12.3 14.7   .  15.1 13.5
85 70  14.8 15.6 14.8 15.9 16.0
85 80  14.3 13.9 15.5 15.6 15.3
85 90  13.3   .  13.4 12.7 12.5
;;;
```

The final example demonstrates a more explicit form of looping in order to simulate a random sample of 50 normal data.

```
data DATASET5;
input MEAN STDDEV;               * Input parameters;
do I=1,50;                       * Begin loop;
   X=MEAN+STDDEV*normal(0);      * normal(0) is a std. normal r.v.;
   output; end;                  * Output X and end loop;
drop MEAN STDDEV;
cards;
40 15
;;;
```

3. Procedures

The most commonly used statistical procedures are given below with the required syntax and primary options. This is meant to provide basic useful information and is not at all comprehensive. Consult the manuals for more options and examples.

ANOVA: Analysis of Variance.
For balanced experimental designs, ANOVA is the most efficient analysis, but it can not produce residuals (see instead GLM, or INSIGHT in Section 6). The syntax is

```
proc anova data=dataset; class variables;
   model response variable = factors and interactions;
   means effects/options;
```

For example, if `DATASET4` shown above did not have missing data the following could be used.

```
proc anova data=DATASET4;
   class TEMP HUMIDITY;
   model STRENGTH=TEMP|HUMIDITY; * Factorial model with interaction;
   means TEMP/duncan;            * Duncan's test on TEMP means;
```

All of the factors in the `model` statement must appear in the `class` statement. The interactions can be given explicitly or implicitly. Examples include:

```
model Y=A B;                * Factors A and B, no interaction;
model Y=A B A*B;            * Factors A and B, and interaction A*B;
model Y=A|B;                * Factors A and B, and interaction A*B;
model Y=A|B C;              * Factors A, B and C, interaction A*B;
model Y=A B(C);             * Factors A and B, and C within B;
```

The effects in the `means` statement can be any effect included in the model statement. Means and standard errors are given for each main effect level and for each crossed effect level. Multiple comparisons are performed for the main effects according to the option requested. Options for the `means` statement include:

`bon`	Bonferroni test
`duncan`	Duncan's multiple range test
`lsd`	Fisher's least significance difference test (default)
`tukey`	Tukey's studentized range test

CHART: Frequency Bar Charts.
This provides a quick histogram-like chart, with more flexibility than the stem-leaf plots of UNIVARIATE. (See also INSIGHT in Section 6.) For eight or more intervals, the horizontal plots are usually better. The syntax is

`proc chart data=`*dataset*`; hbar` *variables*`/`*options*`;`

For example,

```
proc chart data=DATASET4;
   hbar STRENGTH/type=pct midpoints=12 to 20 by 2;
```

Use `vbar` instead of `hbar` for a chart with vertical bars. Options to determine how the scale of the bars is determined include

`axis=`*value*	maximum value of the bar axis
`type=freq`	bar length represents the frequency (default)
`type=cfreq`	bar length represents the cumulative frequency
`type=pct`	bar length represents the relative frequency
`type=cpct`	bar length represents the cumulative relative frequency

By default SAS will choose the categories or the intervals for the chart. To specify them instead use the option

`midpoints=`*values*

If *values* are numerical then the interval endpoints are chosen halfway between the specified values. Equally spaced values result in a chart with equal length intervals (except possibly the end intervals). The values can also be nominal values of a categorical variable. Examples are

`midpoints = 1 3 5 7 9;`	intervals are [*min.*,2), [2,4), [4,6), [8,*max.*)
`midpoints = 1 to 9 by 2;`	same as above
`midpoints = 1 3 7 15;`	intervals are [*min.*,2), [2,5), [5,11), [11,*max.*)
`midpoints = A B C;`	categories are A, B and C

Side-by-side charts can be given for each subgroup defined by a group variable by using the option

`group=`*variable*

CORR: Correlation Matrix.
For multivariate data, the sample correlations are provided as well as the sample means and standard deviations of each variable. The syntax is

`proc corr data=`*dataset options*`; var` *variables*`;`

Options include

`pearson`	Pearson or usual correlation (default)
`spearman`	Spearman rank correlation

FREQ: Frequency Table.
This tabulates frequencies and/or relative frequencies according to categorical variables given by the user. It is especially useful for cross-tabulation (when there are two or more categorical variables) and it can do the chi-square contingency or homogeneity test. The command is given by

```
proc freq data=dataset; weight frequency variable;
   var variables;
```

The weight or frequency variable is used when the dataset consists of counts for each of the subclasses, rather than the individual observations. To do the contingency test or the homogeneity test use

`tables` *class variable*`*`*class variable*`/`*options*`;`

For example,

```
proc freq data=DATASET4; var TEMP HUMIDITY;
   tables TEMP*HUMIDITY/expected chisq;
```

The options include

`cellchi2`	chi-square term for each cell
`chisq`	Pearson's chi-square test
`expected`	expected frequency for each cell under null hypothesis

GLM: General Linear Models.
This conducts the analysis of variance for general linear models, including those for experimental designs and models with covariates. GLM should be used instead of ANOVA when the experimental design is not balanced. GLM can also be used for multiple regression but usually REG is the better procedure. Residuals can be obtained for a residuals analysis. (INSIGHT can perform the basic analyses of GLM interactively, but not the multiple comparisons or special tests.) The syntax for GLM is

```
proc glm data=dataset; class classification variables;
   model response variable = predictor variables / options;
   output out=new dataset keyword = name;
```

The predictor variables can be either numerical or class variables, but if any are the latter they must be identified in the `class` statement. Interactions may be included as in ANOVA (see ANOVA above). Additionally, quadratic and interactive terms with numerical variables may be included as, e.g., `X*X`, `A*X` and `X*Y`.

Additional statements which may used in GLM, but which must appear after the model statement, include the `means`, `lsmeans`, `estimate` and `contrast` statements. These are discussed in more detail below.

We provide two examples. The first involves an experimental design with unbalanced data (i.e., unequal sample sizes).

```
proc glm data=DATASET4; class TEMP HUMIDITY;
   model STRENGTH=TEMP|HUMIDITY; * Factorial model with interaction;
   lsmeans TEMP|HUMIDITY/stderr tdiff;
                                 * Compare all cell means;
```

The second example is a model with both categorical and numerical variables. This situation is sometimes called covariate analysis or regression with unequal slopes.

```
proc glm data=DATASET1; class SEX;
   model WT=SEX HT SEX*HT/ss3 solution;
      * The SEX*HT term allows for the slope to depend on SEX;
   output out=DATANEW r=RESIDUAL p=PREDICTD;
   estimate 'Avg Male Wt at 70 In' SEX HT SEX*HT 0 1 70 0 70;
      * The coefficients for SEX are 0 (Female) and 1 (Male);
   estimate 'Diff Avg Wt at 70 In' SEX HT SEX*HT 1 -1 70 70 -70;
      * The coefficients for SEX are 1 (Female) and -1 (Male);
```

Options for the `model` statement include

`i`	prints the inverse of the crossproducts matrix, $X'X^{-1}$
`noint`	suppresses the intercept term (default is to include it)
`solution`	prints parameter estimates
`ss1`	prints the sequential sums of squares, according to the sequence of predictors

`ss3`	prints the partial sums of squares for each predictor
`xpx`	prints the crossproducts matrix, *X'X*

The options `ss2` and `ss4` are for certain exceptional situations and generally should be avoided.

Certain values may be produced by the `output` statement and appended to the dataset (thus forming a new dataset). These are indicated by *keyword* above and include

`p`	values predicted by the fitted model
`r`	residuals from the fitted model

For experimental designs the population means are estimated with the `means` and/or `lsmeans` statements. Their syntax is

```
lsmeans effects / options;
means effects / options;
```

The options for the `means` statement are the same as those for ANOVA (see ANOVA). Generally, if the experimental design is not balanced then the `lsmeans` statement should be used. The options for `lsmeans` include

`pdiff`	*p*-values for all pairwise comparisons
`stderr`	standard errors of the estimates
`tdiff`	*t*-statistics and *p*-values for all pairwise comparisons

Specific hypothesis tests and estimates are obtained with the `contrast` and `estimate` statements, respectively. The syntax is

```
contrast 'label' effects values;
estimate 'label' effects values;
```

In these statements the *label* should be no more than 20 characters. For a class variables (i.e., a factor) the *values* are a list of coefficients, one for each level of the factor, in ascending order of the levels. For numerical variables the *values* are usually particular values for those variables. The rules are somewhat complicated for handling *effects* which are not included in the statement so it is often best just to include them all, with zeros as needed. (See the second example above.)

LOGISTIC: Logistic Regression.
This procedure fits logistic regression models to a categorical response variable. Logistic regression can be quite sophisticated, depending on the assumptions. We just discuss the bare bones here. Note that the response is assumed to be from ordered categories. The model fits the *cumulative* probabilities for those categories. If the category labels are not in alphabetic or numeric order then the data should be grouped by category and ordered appropriately. The syntax for LOGISTIC is as follows

```
proc logistic data=dataset options;
   model response variable = predictor variables / options;
   output out=new dataset keyword = name;
```

In the `proc logistic` statement options include

`descending`	use if the categories are in reverse order (recommended if the response is binary)
`notsorted`	use when response categories are not in alphanumeric order and SAS should not sort them
`simple`	prints simple statistics for each predictor variable

Options in the model statement can include

`cl`	provide confidence limits for parameters
`corrb`	print correlation matrix of parameter estimates
`covb`	print covariance matrix of parameter estimates
`ctable`	classification table if the response is binary
`noint`	suppress intercept in the model
`risklimits`	provide confidence limits for odds ratios
`rsquare`	provide *R*-square of model fit

Keywords allowed in the output statement include

`p`	predicted *cumulative* probability for the response category
`reschi`	residual based on Pearson's chi
`resdev`	deviance residual

There are many other options available.

MEANS: Summary Statistics.
This procedure calculates summary statistics, such as sample mean and variance. It differs from UNIVARIATE in that the resulting statistics may be output to a new dataset. The syntax is

```
proc means data=dataset keyword list; var variables;
   class classification variable;
   output out=dataset keyword(s)=name(s);
```

The keywords in *keyword list* are names for various summary statistics such `N`, `MEAN`, `STD`, `MIN`, `MAX` and `VAR`. These same keywords are used in the `output` statement to assign names to the variables in the output dataset. For example,

```
proc means data=DATASET4 n mean std; var STRENGTH;
   class TEMP HUMIDITY; out=DATANEW n=N mean=MEAN std=STD;
```

NPAR1WAY: Kruskal-Wallis Test.
Distribution free procedures for the one-way classification model such as the Kruskal-Wallis test are obtained by

```
proc npar1way options data=dataset;
   var response variable; class classification variable;
```

The Kruskal-Wallis test is also called the Wilcoxon test when there are two groups. To use this test to compare weights for the two sexes in DATASET1,

```
proc npar1way wilcoxon data=DATASET1; var WT; class SEX;
```

Options include

`anova`	the usual one-way analysis of variance (*F*-test)
`wilcoxon`	the Kruskal-Wallis test

Other options may be used, depending on assumptions.

PLOT: Character Plot.
Although the resolution is not high, reasonable scatterplots can be obtained. (See also INSIGHT in Section 6.) The syntax for PLOT is

```
proc plot data=dataset;
   plot Y variables * X variables = plot variable/options;
```

The plot variable is optional. The default will be a letter indicating the number of data plotted at that location (`A` for one observation, `B` for two observations, etc.). The plot variable can be the value of a variable in the dataset or it can be a single character in quotes, e.g. `'*'`. Options for PLOT include

`overlay`	to overlay two or more plots
`haxis = values`	identify values for the horizontal (*X*) axis
`href = value`	identify where to draw the vertical (*Y*) axis
`vaxis = values`	identify values for the vertical (*Y*) axis
`vref = value`	identify where to draw the horizontal (*X*) axis

Some examples are

```
proc plot data=DATASET1;
 plot WT*HT/haxis=60 to 75 by 5; * plot of WT vs. HT;
proc plot data=DATASET1;
 plot WT*HT='*';                 * plots with character *;
proc plot data=DATASET1;
 plot WT*HT=SEX;           * plot character is the value of SEX;
proc plot data=DATASET4;
 plot STRENGTH*(TEMP HUMIDITY);
                           * two plots with different X variables;
```

```
proc plot data=DATASET;
 plot Y1*X='*' Y2*X='+'/overlay; * overlays two plots;
```

PRINT: Listing of Data.
This lists each observation with its values for the variables specified. NOTE: This procedure *does not* send output to the printer. The syntax is

```
proc print data=dataset; var variables;
```

If no variables are provided then all the variables are output by default.

RANK: Data Ranking.
The data ranked in order of value (alphabetic order for character data). One option will produce the corresponding quantiles of the normal distribution. SAS creates a new dataset with all the old variables plus the rank values of each ranked variable. To rank the data use

```
proc rank data=dataset out=new dataset;
   var variables; ranks ranked variables;
```

For example

```
proc rank data=DATASET1 out=DATANEW;
   var HT WT; ranks HT_RANK WT_RANK;
```

To obtain normal quantiles (called `N_QUANT` here) for the variable `X` and then to plot `X` in a normal quantile plot use

```
proc rank data=DATASET out=DATANEW normal=blom;
   var X; ranks N_QUANT;
proc plot data=DATANEW; plot X*N_QUANT;
```

REG: Regression Analysis.
Estimates of the coefficients of a multiple regression and their standard errors are provided. It also provides an analysis of variance. Diagnostics for residuals can be obtained. (INSIGHT will do regression interactively, except for the model selection.) The syntax is

```
proc reg data=dataset;
   model response variable = predictor variables/options;
   output out=new dataset keyword(s)=name(s);
```

For example,

```
proc reg data=DATASET4;
   model STRENGTH=TEMP HUMIDITY; * Regression with two predictors;
   output out=DATANEW r=RESIDUAL p=PREDICTD;
```

Each predictor variable must have been defined in a datastep. For example, `X*X` cannot be used (as it can in GLM) to indicate a quadratic term. Options for the `model` statement include

`i`	prints the inverse of the crossproducts matrix, $X'X^{-1}$
`cli`	confidence limits for predicting individual values
`clm`	confidence limits for estimating the regression curve (mean)
`noint`	suppresses the intercept term (default is to include it)
`r`	requests an analysis of the residuals
`ss1`	prints the sequential sums of squares, according to the sequence of predictors
`ss2`	prints the partial sums of squares for each predictor
`vif`	prints the variance inflation factor for each parameter estimate
`xpx`	prints the crossproducts matrix, $X'X$

Certain values may be produced by the `output` statement and appended to the dataset (thus forming a new dataset). These are indicated by *keyword* above and include

`p`	values predicted by the fitted model
`r`	residuals from the fitted model

Model selection criteria can also be obtained from `proc reg`. Note, however, that the above options cannot be used in conjuction with the selection option. The syntax is

```
proc reg data=dataset;
   model response variable = predictor variables/selection=rsquare options;
```

Options include

`adjrsq`	adjusted *R*-squared criterion
`aic`	Akaike's infomation criterion
`b`	provide parameter estimates
`best=`*number*	list only the best (by *R*-square) models for each given number of parameters
`cp`	Mallow's C_p criterion
`include=`*number*	consider only models containing at least the first *number* of predictors
`rmse`	root mean squared error criterion
`sse`	sum of squared errors criterion

RSQUARE: Regression Model Fits.
This procedure calculates model fit for full and reduced models according to various specified criteria. It is essentially the same as the selection procedure in REG. Use

```
proc rsquare data=dataset;
   model response variable = predictor variables/options;
```

For example

```
proc rsquare data=DATASET4;
   model STRENGTH=TEMP HUMIDITY/adjrsq cp rmse;
```

The model statement is the same as for `proc reg` (see REG). The options are also the same, except for `best`.

SORT: Data Sorting.
The data are sorted alphabetically according to or by order of value for one or more variables. This procedure is required before using the `by` statement in other procedures to ensure the data are properly sorted. Use

```
proc sort data=dataset; by option variable ... option variable;
```

If more than one variable is specified the dataset is ordered by the first variable first then the second, etc. The default is to sort in ascending order but any variable name may be proceeded by the option

`descending`	to sort in descending order

TTEST: Student's Two Sample *t*-test.
This does the usual *t*-test to compare means from two independent samples. The command is given by

```
proc ttest data=dataset; class classification variable;
   var variables;
```

The classification variable is required and must consist of exactly two levels. To compare the average weight of males and females in DATASET1,

```
proc ttest data=DATSET1; class SEX; var WT;
```

UNIVARIATE: Summary Statistics.
Summary statistics are calculated, including moments and percentiles. The information is strictly univariate and no multiple variable analysis is provided, except that the analysis can be broken down for categories by using a by statement. The syntax is

```
proc univariate data=dataset options; var variables;
```

Options include

`normal`	provides a test of normality
`plot`	provides a stem-leaf plot, a box-plot and a normal plot
`def=5`	the best definition for sample percentiles

The plots provided by UNIVARIATE are not particularly good because they are quite small. But if a `by` statement is used then side-by-side box-plots will be provided on a full page. Larger histograms can be obtained with CHART and normal plots can be produced using normal quantiles obtained with RANK (see RANK).

VARCOMP: Variance Components Analysis.
This procedure analyzes models with random effects (i.e., with factors having random values for levels).

```
proc varcomp data=dataset method=type1;
   class variables;
   model response variable = effects / option;
```

The *effects* are factors and interactions comprised from the `class` variables. There is only one option.

`fixed=`*number* the first *number* effects are fixed effects, the remainder are random

4. Other Commands

BY: Subgroup Analysis.
Most SAS procedures can be performed on subgroups of the dataset if the subgroups are defined by a classification variable and if the dataset is first sorted by that variable. Use the statement

```
by option variable ... option variable;
```

The class variables must appear in the same order as in the most recent `sort` procedure and the option `descending` must be used or not used as in the `sort` procedure. For example to get two scatterplots, according to `SEX`,

```
proc sort data=DATASET1; by SEX;
proc plot data=DATASET1; by SEX; plot WT*HT='*';
```

Or to obtain summary statistics according to `TEMP` and `HUMIDITY` and side-by-side box-plots,

```
proc sort data=DATASET4; by TEMP descending HUMIDITY;
proc univariate data=DATASET4 plot; var STRENGTH;
   by TEMP descending HUMIDITY;
```

LABEL: Variable Labels.
Variables may be labeled with a character string (in quotes) at any time. If the labeling occurs in a datastep the labels will apply henceforth; otherwise they apply only to the current procedure. The syntax is

```
label variable = 'character string' … variable = 'character string';
```

For example,

```
label NAME='Name of Patient'  WT='Weight at Admission'
      HT='Height at Admission';
```

OPTION: Page Options.
The statement to set up page options is

```
option options;
```

and possible options include

`center`	center title, if any
`date`	include job execution date on first line
`ls=`*number*	line length is *number* characters
`nocenter`	do not center title, if any
`nodate`	do not include job execution date on first line
`ps=`*number*	page length is *number* lines

RUN: Run Statement.
The `run` statement tells SAS that all commands for a particular procedure have been provided and is required at the end of a sequence of commands submitted interactively. The syntax is

```
run;
```

TITLE: Page Titles.
Page titles may be provided and changed at any time. They may have multiple lines. For the first line of a title (given in quotes) use

```
title 'title';
```

and for additional lines use

```
title# 'title';
```

where # is the line number on which the title is to appear.

5. Operators, Expressions and Functions

OPERATORS: The usual mathematical operators `+`, `-`, `*`, `/` and `**` (for exponentiation) are available as well as the comparison operators `=`, `>`, `<`, `>=`, `<=`. Also, `not`, `and` and `or` may be used in logical expressions.

EXPRESSIONS: Expressions are constructed according to the usual rules of syntax in most programming languages such as C or Fortran. Logical expressions are given the numerical values 0 (false) or 1 (true) and may be used as such.

FUNCTIONS: Standard functions are available as well as many statistical functions. The standard functions include `abs`, `max`, `min`, `sqrt`, `exp`, `log` (natural logarithm), and `log10` (logarithm base 10). Trigonometric and hyperbolic functions include `sin`, `cos`, `tan`, `arsin`, `arcos`, `atan`, `sinh`, `cosh` and `tanh`.

Function that compute sample statistics either take the form *function*`(`*list*`)` where *list* is a list of values and variables (e.g., `sum(1,2,X,Y,Z)`) or take the form *function* `(OF` *list*`)` where *list* is a list only of variables, separated by spaces, and may be less explicit (e.g., `sum(OF X1-X10 Y1-Y5)`). These functions include the following.

`n`	no. of nonmissing arguments	`nmiss`	no. of missing arguments
`sum`	sum	`mean`	average
`min`	smallest value	`max`	largest value
`std`	standard deviation	`var`	variance
`skewness`	skewness	`kurtosis`	kurtosis
`cv`	coefficient of variation		

Among the statistical functions are those which give the cumulative probability function for various distributions:

`probnorm(Z)`	standard normal	`probt(T,DF`	Student's *T*
`probchi(X,DF)`	chi-square	`probf(F,NDF,DDF)`	Fisher's *F*
`probgam(X,ALPHA)`	gamma		

The inverses to these functions (which compute percentiles) include:

`probit(P)`	standard normal	`tinv(P,DF)`	Student's *T*
`cinv(P,DF)`	chi-square	`finv(P,NDF,DDF)`	Fisher's *F*
`gaminv(P,ALPHA)`	gamma		

There are also functions to generate pseudo-random variables: `normal(SEED)`, `uniform(SEED)` and `rangam(SEED,ALPHA)`. The argument `SEED` must be a constant, either 0 or a 7-digit odd integer and is used only the first time the function is evaluated. If `SEED` is 0 then a function of the current time is used for a seed.

6. Interactive Data Analysis (INSIGHT)

Most of the basic model fitting and testing (as in REG or GLM) can be done interactively with SAS using INSIGHT, which is windows oriented and menu driven. It is also the best and easiest way to get decent graphs. Before it can be used you first must create the desired data (with a datastep or as output from a procedure).

To start INSIGHT select Analyze-Interactive Data Analysis from the Globals menu. This brings up a menu of data libraries. Choose WORK and then the dataset desired. SAS then provides a spreadsheet of the data. (It is also possible to enter data into the spreadsheet.) By double clicking on a variable in the spreadsheet, you bring up a dialog box for that variable. You can then change or assign labels, variable type and other attributes.

Now choose the desired analysis from the Analyze menu. To select the variables for the analysis, highlight the variable names and then click on Y or X as desired. In addition, each analysis has a number of options concerning the type of plots to be produced. Both DISTRIBUTION and FIT analyses include options for "smooth" fitting of the data. You can also obtain residuals from the FIT analysis and perform residuals diagnostics.

Note that the output is graphical. Tables and graphs should be saved separately as follows. To save tables, hold the Ctrl key down while highlighting each table you want to save. Select File-Save-Tables, which copies the tables to the OUTPUT window. Select the OUTPUT window, then select File-SaveAs to save the output as a text file (e.g., *file.txt*) which may be edited later. You cannot edit in the OUTPUT window. To save a graph, first highlight it. Then select File-Save-GraphicsFile and check GIF and grey scale (unless you really want color). Do not save it as a bitmap file because the file will be 10-100 times larger with no improvement in quality. Give a filename (e.g., *file.gif*) and click OK. The graphics file may be inserted as a picture with a suitable word processor. Repeat the above for each graph you want to save. (Except when graphs are presented adjacent to each other, it is recommended you save each graph separately.)

For more information, see the Help menu while in an INSIGHT window. Specifically, Help-References and Help-Techniques have information about the analyses available in INSIGHT.