Introductory Guide to SAS:

For UVM Statistics Students

By Richard Single

Contents

1 Introduction and Preliminaries

- SAS is composed of a collection of procedures (PROCs). These operate on data sets created using the DATA Step described below.
- SAS command files should end with the .sas extension (e.g., *example1.sas*). SAS will then recognize it as a command file and you can open it by double-clicking on the file.

  SAS opens with a task bar at the bottom, similar to the Windows task bar. This usually contains three windows: Editor, Log, and Output.

  You can edit a .sas command file in the SAS editor or in any text editor (Word, Notepad, etc.).
- Commands are submitted to SAS while in the Editor window either by selecting *Run : Submit* or by clicking on the running icon.

  By default, SAS executes all commands in the Editor window. You can highlight/select a subset of commands before submitting them in order to restrict the execution to this subset.

  If there are no errors output will appear in the Output window and a listing of the time and commands executed will appear in the Log window.

  If there are errors, a message will be printed in the Log window and output may not be created. **It is always a good idea to check the Log window for error and/or warning messages.**

  You can save the results of any of these windows by selecting *File : Save As*
- **Every command line in a SAS program must end with a semicolon;**
- There are two ways to add a comment to (or comment out a section of) a program:
  - An asterisk at the beginning of a line comments out everything until the next semicolon.

    * Commented Text ;
  - Any text inbetween the following set of characters is a comment.

    /* Commented Text */
- Note: In order that your output will print on a single page the following command should appear as the first line of any SAS program:

  OPTIONS LINESIZE = 75 PAGESIZE = 64 NODATE;

  This specifies that each line of output will contain at most 75 characters and that there will be no more than 64 lines per page.

2 Reading in Data: The DATA Step

- In what follows, anything in *italics* is a user specified name.
- Anything inside of < > indicates an optional part of a SAS command. It is not required in order for the command to be executed.

DATA *Dataset_name* ;

INFILE *'datafile-name'* <FIRSTOBS = 1 > ;

INPUT *variable_names* ;

RUN ;

2.1 The DATA Statement

- This statement creates a SAS data set with the name *Dataset_name*. **Dataset names can not include a dash ("-")**. Use an underscore ("_") instead of a dash ("-").

2.2 The INFILE Statement

- This statement informs SAS where the Data file is located. The name of the data file must be enclosed either in single (') or double quotes ("). SAS is very picky about this: you must use ' rather than ‘ to enclose the name of the file.
- SAS assumes that the data file is in the Current Working Directory (CWD). The CWD is listed under the SAS task bar on the bottom right next to the hard-drive icon. You can change the directory by double-clicking on the icon and selecting a new directory.

  Alternatively, you can specify the complete path to the file, such as,

  ```
  "C:\Documents and Settings\Rich\My Documents\projects\height.sas"
  ```

- Options:
  - FIRSTOBS=n This tells SAS to look for the first observation on the nth line of the data file. This is convenient if there is a header with variable names in your data file. If no FIRSTOBS statement is given then SAS looks for the first observation on the first line of the data file.

2.3 The INPUT Statement

- This statement names the variables and also tells SAS what the format of the data file is.
- **Variable names may be 1 to 32 characters in length, must begin with a letter, and can not include a dash ("-").** Use an underscore ("_") instead of a dash ("-").
- The format of the INPUT statement is as follows:
  - Variables are listed in the order in which they appear in the data file.
  - A $ must follow the name of any variable with observations that contain alphabetic characters.
  - If there are no spaces between columns of observations for different variables, then you must specify the column locations for each variable.
  - Note: Spaces between columns of observations must be actual spaces, not tabs (unless you include EXPANDTABS on the INPUT statement). SAS does not recognize tabs unless you tell it to do so.

2.4 An Example

Suppose that the file *height.dat* has data on the Name, Gender, Age, and Height of 6 individuals. The first line of the file has the variable names, thus the first observation is on line 2. The Name variable occupies columns 1 through 5; Gender is in column 7; Age is in columns 9 and 10; and Height is on columns 12 and 13.

```
---------------
name  gender age height
John  M      11  51
Jane  F      13  62
Jim   M      15  58
Judy  F      15  61
Joyce F      12  55
Jeff  M      16  66
---------------
```

The following Data step will read this data into the Data set named A (assuming that the file *height.dat* is in the current working directory).

```
DATA A;
  INFILE 'height.dat' FIRSTOBS = 2;
  INPUT name $ 1-5 gender $ 7 age 9-10 height 12-13;
RUN;
```

Note: In this case the INPUT statement could have been written without the column numbers since there is a space between each of the observations:

```
INPUT name $ gender $ age height;
```

2.5 Defining New Variables

Suppose that in the previous example you wanted to create two new variables: $\log(height)$ and age^2.

- New variables can be created when the data is first read into SAS.

  These must be defined after the INPUT statement as in the following:

```
DATA A;
  INFILE 'height.dat' FIRSTOBS = 2;
  INPUT name $ gender $ age height;
  w1 = log(height);
  w2 = age**2;
RUN;
```

- Alternatively, new variables can be defined using another DATA Step that "recycles" the data from the original Data Step, using the SET command.

```
DATA B;
  SET A;
  w1 = log(height);
  w2 = age**2;
RUN;
```

  Note: Data set B contains all of the data in data set A as well as the two new variables $w1$ and $w2$.

3 PROC PRINT

PROC PRINT DATA = *Dataset-name*;

< **VAR** *variable(s)* ; >

- This procedure prints out a specified data set.
- The optional VAR statement specifies which variables will be printed and the order in which they will appear. If no VAR statement is given then all variables are printed.

4 PROC PLOT

PROC PLOT DATA = *Dataset-name* < VPERCENT = 50 > ;

PLOT *y-variable* * *x-variable*;

- This procedure creates a scatter plot of the variables named in the PLOT statement.
- Several PLOT statements can be made (each on a separate line).
- *y-variable* is the name of the variable to appear on the Y-axis. Similarly for *x-variable*.
- The optional VPERCENT command reduces the height of the plot (it controls the Vertical Percentage of a page that the plot occupies). A value of 50 percent usually works well.

5 PROC SORT

PROC SORT DATA = *Dataset-name*;

BY *variable(s)*;

- This procedure sorts the specified data set by the variable(s) in the BY statement.

6 PROC MEANS

PROC MEANS DATA = *Dataset-name* < *options* > ;

VAR *variable(s)*;

< **BY** *variable(s)*; >

RUN;

- This procedure gives the following summary statistics (if no options are specified) for each variable listed in the VAR statement: number of observations, mean, standard deviation, minimum value, maximum value, and standard error of the mean.
- Options: You can select which statistics you want to have printed from the following list: N, NMISS (number of missing values), MEAN, STD, VAR, MIN, MAX, SKEWNESS, KURTOSIS, T (Student's t statistic for testing the hypothesis that the population mean is zero), PRT (the p-value for the two-sided test that the population mean is zero)

7 PROC TTEST

7.1 Some Preliminary Information

- This procedure performs 2 sample t-tests of the null hypothesis that the mean of two groups are equal. Observations in each group are assumed to have been sampled from independent normal distributions.
- Equal-variance and Unequal-variance t-tests are performed as well as a test of the null hypothesis that the variances are equal. Thus, it is important to check this later test before deciding which t-test to consider unless you have further information suggesting one over the other.

PROC **TTEST** DATA = *Dataset-name* ;

CLASS *variable* ;

< **BY** *variable(s)* ; >

< **VAR** *variable(s)* ; >

RUN ;

7.2 The CLASS Statement

- This statement specifies the name of the grouping variable indicating which 2 groups are to be compared.
- The grouping variable must have exactly 2 levels and can be of either character or numeric type.

7.3 Optional Statements

- The **BY** Statement
  - Separate analyses on observations in the two groups are given for each level of the variable(s) specified in the BY statement.
  - Note: The data set must first be sorted by the variables in the BY statement (A PROC SORT must come before the PROC TTEST if this is the case).
- The **VAR** Statement
  - The means are compared for all variables in the VAR statement.
  - If no VAR statement is given, then analyses are done for all numeric variables in the data set (excluding the grouping variables).

8 PROC GLM for the Analysis of Variance

8.1 Some Preliminary Definitions and Notation

1. An *effect* is an independent variable that appears in the MODEL statement. It may be a single independent variable listed in the CLASS statement or an interaction between two or more independent variables listed in the CLASS statement (e.g., If we model Y as a function of a and $a * b$, then both a and $a * b$ are effects).
2. Anything inside of < > indicates an optional part of a SAS command. It is not required in order for the command to be executed. Typically these are options which occur at the end of a command following a forward slash.
3. Anything in *italics* is a user specified name (consistent with those in the DATA Step).

PROC **GLM** DATA = *Dataset-name* ;

CLASS *variables* ;

MODEL *dependent-variable* = *effects* < / options > ;

CONTRAST *'label'* *effect* *values* ;

MEANS *effect* < / options > ;

LSMEANS *effect* < / options > ;

RANDOM *effect* / TEST ;

TEST < H = *effects* > E = *effect* < / options > ;

OUTPUT OUT=*newdata* RESIDUAL=*resid* PREDICTED=*pred* STUDENT=*stresid* ;

RUN ;
QUIT ;

8.2 The CLASS Statement

- This statement names the classification variables that will be used in the analysis (e.g., *trtment, gender, group, rep*). The classification variables can be of either character or numeric type.
- When there are more than one classification variables, the order that they are listed sets the parameterization for these terms in contrasts and model matrices (see the CONTRAST statement below).

8.3 The MODEL Statement

- This statement names the dependent variable and independent effects.
- Options:
  - CLM this produces 95% confidence limits for a mean predicted value for each observation.
  - P this prints predicted, observed, and residual values for each observation that does not have missing values for independent variables.
  - SOLUTION this prints the parameter estimates for the effects in the model.

8.4 The CONTRAST Statement

- This statement allows you to perform customized hypothesis tests about the parameters (say $\beta_1, ..., \beta_k$). This is done by specifying a vector (or Matrix) $\mathbf{L}$ in order to test $H_0 : \mathbf{L}\ \beta = 0$, where $\beta = (\beta_1, ..., \beta_k)^T$.
- Several contrast statements can be used (each listed separately).
- A *label* of up to 20 characters is required to identify each contrast. It must be enclosed in single quotes.
- The *effect* specifies the variable of interest (it must appear even if there is only one variable in the model).
- *values* are the elements of the vector $\mathbf{L}$ used to form the specific hypothesis test.
- Options:
  - E = *effect* this specifies an error term to be used as the denominator of the F test (it must be an effect in the model). The default error term is the residual or error mean square.

- An Example:

  Suppose that you are fitting an ANOVA model for the dependent variable Y as a function of the variable A which has 4 levels.

  The vector of parameters for the effect A in this model is $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$.

  To test $H_0 : \alpha_1 = \alpha_2$ the following vector **L** would be used: **L** = (1,-1,0,0)

  Note: SAS does not accept commas between the elements of the vector (see example below). Only decimal entries are allowed.

  This corresponds to the contrast given in the code below:

```
PROC GLM DATA = example;
  CLASS A;
  MODEL Y = A;
  CONTRAST 'a1 vs. a2'   A   1   -1   0   0 ;
RUN;      QUIT;
```

8.5 The MEANS Statement

- This statement computes arithmetic means and standard deviations of the *dependent-variable* for each level of the *effect(s)* specified. *Effects* must be variables listed in the CLASS statement (CLASS variables).
- Several MEANS statements can be listed.
- Options:
  - LSD performs Fisher's Least Significant Difference test (also known as Fisher's Protected T-test). These are displayed as pairwise *t*-tests.
  - SCHEFFE performs Scheffe's multiple comparison test for the means of all main effects in the means statements.
  - LINES requests output for multiple comparisons as ranked means with LINES under values that are not significantly different. This is the default.
  - CLDIFF requests output for multiple comparisons as Confidence Limits for DIFFerences (i.e., confidence intervals).
  - ALPHA=.01 specifies an $\alpha = .01$ significance level. Any value between .0001 and .9999 may be used. The default value is .05
  - E = *effect* this specifies an error term to be used as the denominator of the F test (it must be an effect in the model). The default error term is the residual or error mean square.
  - HOVTEST=BF specifies Brown and Forsythe's variation of Levene's test, sometimes referred to as "Levene's Median" test (Brown and Forsythe, 1974).

8.6 The LSMEANS Statement

- This statement computes least-squares means (LS-means) for each effect listed in the LSMEANS statement. LS-means are adjusted for other variables in the model that estimate the marginal means over a balanced population.
- Several LSMEANS statements can be used and only CLASS variables can be listed in the LSMEANS statement.
- The LSMEANS statement performs multiple comparisons on interactions as well as main effects.
- Options:

- ADJUST=BON (or DUNNETT, SCHEFFE, TUKEY) perform the corresponding multiple comparison procedures. BON gives the Bonferroni adjustment. TUKEY gives the Tukey-Kramer adjustment when data are unbalanced. For DUNNETT's method, you should specify PDIFF=control('*level*') [see below] since the default is to use group1 as the control.
- ALPHA=.01 specifies an $\alpha = .01$ significance level. Any value between .0001 and .9999 may be used. The default value is .05
- CL requests Confidence Limits for the individual LS-means.
- PDIFF requests P-values for DIFFerences of the LS-means (as a matrix of pairwise values). PDIFF=control('*level*') requests the differences with a control at the specified *level* (variable value).
- E = *effect* this specifies an error term to be used as the denominator of the F test (it must be an effect in the model). The default error term is the residual or error mean square. ETYPE=n optionally specifies the type (1, 2, 3, or 4, corresponding to Type I, II, III, and IV tests).
- SLICE = *fixed-effect* tests for differences at each level of the $fixed - effect$ for an interaction term, producing simple effects.
- STDERR gives the standard error of the LS-means.

8.7 The RANDOM Statement

- This statement specifies which *effects* in the model are random and tests hypotheses for each *effect* specified based on the proper Expected Mean Squares for a random effects model.
- NOTE:
  - All other statements in the GLM procedure operate under the assumption that all effects are fixed, even if you use a RANDOM statement.
  - All random interaction effects (i.e., interactions between two or more random effects) must be declared in the RANDOM statement as they are not automatically interpreted as random effects.

8.8 The TEST Statement

- This statement allows you to specify the numerator and denominator of the F test in order to correctly perform a test when there is a non-standard error structure (e.g., in a Split-Plot design).
- Options:
  - H = *effect(s)* specifies the effect(s) to be used as hypothesis (or numerator) effect(s).
  - E = *effect* specifies the effect to be used as the error (or denominator) term.

8.9 The OUTPUT Statement

- This statement allows you to specify a new data set (called *newdata* in the above template) using the command OUT= *newdata*.
- Some of the built-in variables available include the following:
  - RESIDUAL=*resid* creates the variable named *resid* containing the residuals from the model.
  - PREDICTED=*pred* creates the variable named *pred* containing the predicted values from the model.
  - STUDENT=*stresid* creates the variable named *stresid* containing the standardized residuals from the model.

8.10 The QUIT Statement

- Proc GLM can be run interactively. After you specify a model with a MODEL statement and run Proc GLM with a RUN statement, you can execute other statements without reinvoking Proc GLM (but, only one MODEL statement is allowed).
- Additional RUN statements are used to execute any additional statements. Thus, a RUN statements does not end Proc GLM.
- The QUIT statement ends a call to Proc GLM. Alternatively, Proc GLM ends when the next DATA step or PROC is submitted.