

# Intro to SAS and the Data Step

August 24, 2006

## 1 Overview of the SAS system and how to run it

### 1.1 What is SAS?

SAS is many things. Among other things, it can be used for data management and report generation, as a statistical package including descriptive and inferential statistics (for a wide variety of problems) and as a general programming environment. The complete package is extremely comprehensive and we will only touch on the basics here.

### 1.2 Where do I read about things?

There are numerous manuals on using SAS (these make the Bible look like a short story) and many books have been written on specific parts of SAS. A book that might be useful for the beginning part of the course is:

”The Little SAS Book” by Delwiche and Slaughter; SAS Institute.

The many tool I recommend when coming across a problem, is the use of the internet. If any PROC or data problem is googled many examples will be displayed. Also all of the documentation on SAS is now available through the web at:

<http://support.sas.com/onlinedoc/913/docMainpage.jsp>

### 1.3 How do I run SAS?

On a PC, SAS is run in the windows environment. When SAS is opened there are various sub-windows, the three main being the Editor (or program) window, the output window, and the log window. When you are actually using programming statements (as opposed to click and go as in analyst or insight) the programming statements are put in the program window. The job is then submitted (by clicking on the running figure), with output going to the output window and the log window containing information (including error messages) about the job. You can open and save files from any window.

### 1.4 Programming SAS versus ”point and click”

**Using programming statements.**

The **data step** creates a SAS data file. This can be from reading an external file, manually

imputed or a combination of the two. We will focus on mainly reading external files.

Once a SAS data set is in place, there are numerous pre-programmed procedures that can be run on that data set. These are invoked with a **PROC** statement. For the most part, each proc has a fixed set of rules and options that must be followed.

### **Running SAS without programming.**

There are various options now available in SAS for running **PROC**'s without programming, but instead using a "point and click" technique. We will cover certain aspects of this using Analyst/PROC Insight. These are easy to use, but something is lost in this technique.

## **2 Basic elements in the data step.**

### **2.1 Reading from external files.**

We will discuss three main ways:

- Reading space delimited data from a text file.
- Reading comma or tab delimited data from a text file.
- Reading in an excel file.

#### **2.1.1 Reading space delimited data from a text file.**

**When to use:** When there is one or more spaces (it doesn't matter how many) between variables. If there are **missing values** there needs to be an entry indicating the missing value. These should be indicated by a . in the raw text file. Here is part of a sample data set that is on Blackboard called 'Sample Space':

```
1  69.7  3  brown
2  64.7  3  black
3  71.8  2  blonde
....
22 72.7  3  brown
23 65.9  4  blonde
```

This data would be read into SAS using the following statements in the editor:

```
data sample;
infile 'Z:\sample space.txt';
input Unit Height Group Hair $;
proc print;
run;
```

Notice the \$ after Hair. This tells SAS that Hair is not a numeric column. Here is some of the output:

Obs	Unit	Height	Group	Hair
1	1	69.7	3	brown
2	2	64.7	3	black
3	3	71.8	2	blonde
.....				
22	22	72.7	3	brown
23	23	65.9	4	blonde

The data step 'sample' is a temporary data file (Permanent data file's will be discussed later) and will disappear when your session ends. If there is only one data step then all subsequent procs will operate on data sample. Otherwise, the proc will operate on the last data set which was used unless there is a *data = ...* after the proc.

**Note:** For character variables there is a limit of 8 characters and any entry with more than 8 will be truncated. To get around this problem put \$XX. after the character variable, where XX is the length of the longest character string.

### 2.1.2 Reading comma or tab delimited data from a text file.

**When to use:** When there is commas or tabs between variables. Here is a sample of how to run data, that is on Blackboard, called 'Sample Tab' which is tab delimited:

```
data sample;
infile 'Z:\sample tab.txt' dlm='09'x;
input Unit Height Group Hair $;
proc print;
run;
```

Here is a sample of how to run data, that is on Blackboard, called 'Sample Comma' which is comma delimited:

```
data sample;
infile 'Z:\sample comma.txt' dlm=','; /*could use dsd instead of dlm=','*/
input Unit Height Group Hair $;
proc print;
run;
```

Other useful notes while reading in text files:

- **firstobs=n** in the infile command will begin reading at line n of the data file.
- You can abbreviate a list of sequential variables (eg. v1 v2 v3 v4 as v1-v4)
- **title.** The line *title 'string'*; will cause the title string to be printed. At any point in the program a new title line can be used and that title will be printed until the next title line is encountered.
- There are many many options, that you can use. These are a few....

### 2.1.3 Reading in an Excel file.

**When to use:** When reading in data from an Excel spreadsheet. Since an Excel spreadsheet cannot be read into SAS using an infile command, we are forced to use **PROC IMPORT**. Here is an example of importing an Excel file using PROC IMPORT.

```
proc import datafile='Z:\sample dat.xls' out=excel replace;
getnames=yes;
title 'What A Great Lab';
proc print data=excel;
run;
```

- The *getnames=yes* option will take the first line of the spreadsheet as the names of the variables.
- *out=' '* specifies the name of the temporary data set that is created by PROC IMPORT.
- The *replace* option, when displayed will overwrite any existing temporary data set with the name *out=' '*.

And the output:

What a Great Lab				
Obs	Units	Height	Group	Hair
1	1	69.7156	3	brown
2	2	64.6673	3	black
3	3	71.8357	2	blonde
.....				
22	22	72.7067	3	brown
23	23	65.9372	4	blonde

## 2.2 Creating Permanent SAS Files

The SAS data files described above have been created in a temporary environment. These are lost when your session ends unless saved in as permanent SAS data sets. A permanent SAS file will be saved on your drive, and can be recalled in multiple uses.

The libname command simply identifies some name (abc in the example below) with a directory. This name can change.

The first program uses the set command to turn the previous excel file into a permanent SAS file in the 'Z:directory.

```
proc import datafile='Z:\sample dat.xls' out=excel replace;
getnames=yes;
proc print data=excel;
run;
```

```
libname abc 'Z:\';
data abc.perm;
set excel;
run;
proc print data=abc.perm;
run;
```

Within the same session you can just refer to abc.perm. In a later session, a new libname line will recall former permanent data files. The libname (abc above) does not have to be the same, the 'Z:' and perm will have to be the same. The libname simply sets up an equivalence of the name with the directory for the current session.

```
libname usc 'Z:\';
proc print data=usc.perm;
run;
```