Introduction to SAS Procedures and basic Data step programming

Philip Dixon

3 January 2007

This document provides general guidance for writing and running SAS programs and summarizes some of the useful procedures. Examples of these procedures in the context of an entire SAS program are provided on my STAT402 web pages (www.public.iastate.edu/ pdixon/stat402/sasprogs.html).

The examples all use the insect data file (insect.txt), from the data files section of the 402 web page. This file has two measurements on each of 6 trees. The first line is a header line giving variable names. The remaining 6 lines each have data from one tree. There are three pieces of informaiton, the tree number, the number of insect species found last year and the number of insect species found this year.

Values on a line are separated by one (or more) spaces. A lonely . indicates a missing value. SAS can also read files with tabs or commas between fields, but to do this, you need to add options to the INFILE command.

Data steps

INFILE commands, CARDS statements, or DATALINES statements: The easiest way to read data is from a text file. The data can be included with the sas commands or kept separate. You use an infile command when you keep the data separate. You use a cards; or a datalines; statement when you include the data with the sas commands.

The INFILE command specifies the name of the file containing the data. The file name goes in quotes. It can include whatever path information (e.g. 'g:/philip/insect.txt') that is needed.

```
data insect;
  infile 'C:\Documents and Settings\pdixon.IASTATE\My Documents\insect.txt';
  input tree last this;
```

To read data stored with the sas commands, use CARDS; or DATALINES;

```
data insect;
  input tree last this;
cards;
1 42 37
2 51 47
```

```
3 23 17
4 62 .
5 . 32
6 37 27
;
```

Notes: CARDS; (or DATALINES;) is the last command in the data step, the data ends with a ;.

INPUT commands: The input command tells SAS how to read your data. There are many options for reading complex data sets. We'll use the simplest, 'list input'. The input line lists the variables to be read from one line of data. Each data value is separated by one or more spaces. So, the above input lines read three numbers from each line of data. The first is stored in the variable 'tree', the second in 'last' and the third in 'this'.

Character variables: By default, considers all variables as numeric (i.e. containing 1, 2.1, ...). It is often helpful to read character variables (e.g. "Dry", "Moist", "Fluctuating"), especially for treatment or site names. Since "Dry" can not be converted to a number, you need to tell SAS which variables contain character information. You do this by putting a $ after the variable name on the input line. You can not do arithmetic operations (e.g. compute a mean) on character variables.

```
data insect;
  input tree $ last this;
cards;
A 42 37
B 51 47
C 23 17
D 62 .
E . 32
F 37 27
;
```

Missing Data: When you use list input, the order of the values matters. If a value is not recorded (e.g. the last value for tree E), you need to indicate the missing value. SAS uses a lone decimal point (.) to indicate a missing value. You can use this in data sets (e.g. insect.txt) or in commands (see the commands for deleting observations, below).

Saving paper: SAS output (the Listing window) can fill many pages. Two ways to reduce the paper:

1) Add an option to the top of your file:

options formdlim='-';

Pages of output are separated by ———————————- instead of being put on new sheets of paper.

2) Use Microsoft WORD or other software to print 2 or 4 pages on one physical page.

Data step programming

All transformations and computations on variables are done in the DATA step. Each operation is a separate command. Each goes after the input command. Here are some of the common commands:

1. Log transformations (default is natural log): loglast = log(last);
2. Difference: diff = this - last;
3. Recoding (e.g. number to character):
   if tree = 1 then newtree = 'A. rubr';
   else if tree = 2 then newtree = 'B. nigr';
   else if tree < 5 then newtree = 'C. occ';
   else newtree = 'Q. alba';
4. Deleting observations: if tree = 5 then delete;
5. Keeping certain observations (no then clause): if tree > 3;

```
data insect;
  infile 'insect.txt';
  input tree last this;
  diff =  this - last;

proc print;
  title 'Insect data set, with difference';
  run;
```

PROC PRINT: This proc prints out a data set (see example above). The title statement is optional.

PROC PLOT: This proc plots two variables. The plot command specifies the variables to be plotted. The syntax is plot Y * X; The Y axis variable comes first, then the X axis variable.

```
data insect;
  infile 'insect.txt';
  input tree last this;
  diff =  this - last;

proc plot;
  plot this*last;
  title 'Insect data set';
  run;
```

PROC GPLOT: Draws high-resolution plots in a new window. Same syntax as proc plot.

PROC UNIVARIATE and PROC MEANS: Both compute descriptive statistics (means, variances, other quantities). The VAR command specifies which variables are to be used. UNIVARIATE computes more descriptive statistics. MEANS gives you more compact output and is easier to use to produce a new data set. If you want descriptive statistics for subgroups (e.g. for each treatment), then you need to use the BY command. To use BY, the data need to be sorted by that variable first.

The PLOT option to PROC UNIVARIATE draws box plots (low resolution).

```
data insect;
  infile 'insect.txt';
  input tree last this;
  if tree < 4 then group = 'First';
  else group='Second';

proc univariate;
  var last this;
  run;

proc means;
  var last this;
  run;

proc sort;
  by group;
  run;

proc univariate plot;
  by group;
  var last this;
  run;
```

PROC BOXPLOT: Draws high resolution box plots. Only works with windowing SAS. The plot command describes the response variable and the group variable. The syntax is plot response*group;

```
data insect;
  infile 'insect.txt';
  input tree last this;
  if tree < 4 then group = 'First';
   else group='Second';
  run;

proc boxplot;
  plot last*group;
  title 'Box plot of last values for each group of trees';
  run;
```

PROC IMPORT: SAS can import data directly from an EXCEL spreadsheet. The format and types of data that can be read differ slightly in PC and vincent versions. DATAFILE= specifies the data set and its type, indicated by the file extension. .xls is an excel spreadsheet. .csv is a comma delimited file. OUT=specifies the SAS data set name. Here's the PC format. See me if you need to read vincent or MAC files. Again, you may need to use a different path to the file.

```
proc import
  datafile='C:\Documents and Settings\pdixon.IASTATE\MyDocuments\insect.xls'
  out=insect;
run;

proc print;
title 'Insect data set';
run;
```