SAS Tips for data handling

Most of the examples, particularly the ones about the analysis are for users who has some experience with SAS, but as well as familiar with statistics and quantitative genetics. If you are a beginner for SAS, I recommend you to start with some other publications. 'The Little SAS Book, a primer' by Lora D. Delwiche and Susan J. Slaughter is an excellent book for the SAS beginners.

Data Modification Tips

There are many data tips in SAS. Here, I put some of them that
I frequently use.

1. Importing external data files (Excel, Access and delimited). See PROC IMPORT for details.

```
Importing an excel file
Proc import datafile='d:\myfiles\Excel File.xls'
out=mydata replace ;
sheet='sheet1';
run;

Importing Access table
proc import table="test1"
out=mydata dbms=access;
database="c:\myfiles\testdata.mdb";
run;
```

```
Importing an external delimited file
Region,State,Month,Expenses,Revenue
Southern,GA,JAN97,2000,8000
Southern,GA,FEB97,1200,6000
Northern,NY,MAR97,6000,5000

proc import datafile="\myfiles\mydata"
out=mydata dbms=dlm replace;
delimiter=',';
getnames=yes;
run;
```

2. Converting a numeric variable to a character or vice versa

```
Var_cha=put(test, $2.); * to convert to character;
Var_num=input(test,4.1); * to convert to numeric;
```

3. Creating a new variable by compressing several variables

test	family	rep	female	male	tree
1	A	1	F43	M28	1
1	B	1	F22	M02	2
1	B	2	F22	M02	4

```
data a ; set a;
ID=compress(test||family||rep||tree); *Combine factors
using ||, remove any trailing blank using COMPRESS;

CROSSno=trim(female||male); *Combine FEMALE and MALE
variables with ||, remove any trialing blank using TRIM;
CROSSblank=female||male ; * Trim or Compress is not used.
New variable may contain blanks;

Proc print data=a ; run;
Result:
```

test	family	rep	female	male	tree	ID	CROSSno	CROSSblank
1	A	1	F43	M28	1	1A11	F43M28	F43 M28
1	B	1	F22	M02	2	1B12	F22M02	F22 M02
1	B	2	F22	M02	4	1B24	F22M02	F22 M02

4. Creating new variables by partitioning a variable using the SUBSTR

```
* Let tree ID is combination of test#=1, family=A, rep#=2,
tree#=3;
```

ID	vol	dbh	ht
1A23	12.1	4.5	2.5
1A24	14.5	5.2	2.7
1B15	11.8	3.9	2.1

```
data a ;
set a;
tree=substr(ID,4,1); *Use ID, take 4th character to create
TREE number;
family=substr(ID,2,1); * create family code ;
reptree=substr(ID,3,2); *Start from 3rd column, pick 3rd
and the next;
Result:
proc print data=a; run;
```

treeID	vol	dbh	ht	tree	family	reptree
1A23	12.1	4.5	2.5	3	A	23
1A24	14.5	5.2	2.7	4	A	24
1B15	11.8	3.9	2.1	5	B	15

5. Using the TRIM to remove trailing blank

*TRIM copies a character argument, removes all trailing blanks, and returns the trimmed argument as a result;

```
p1=trim(left(p1)); *trim romeves blank, left aligns to left;
p2=trim(left(p2));

* or ;
data a;
input part1 $ 1-10 part2 $ 11-20;
hasblank=part1||part2;
noblank=trim(part1)||part2;
put hasblank;
put noblank;
datalines;

* SAS Statements Results;
x="A"||trim(" ")||"B";
put x;
```

```
Result: A B

x=" ";
y=">"||trim(x)||"<";
put y;
Result: > <
```

6. Removing the blank and aligning LEFT

```
a=' DUE DATE';
b=left(a);
put a;
Result: a='DUE DATE'
```

7. Converting a period separated value to a comma separated value (TRANSLATE)

```
dep=input(translate(depcomma,',','.'),8.3);
```

The Translate statement converts trait **depcomma** from 12.45 to 12,45. Input statement converts it to numeric.

8. Using the AND and OR operators,
You can combine two or more conditions. AND combines two conditions by selecting values that satisfy both conditions, and OR combines two conditions by selecting values that satisfy either or both conditions.

```
Data a ;
set b (keep=ID distance dens ringno type);
*where ID='4N14' | ID ='4N15' | ID ='4N21' ;
where ID in ('4N14', '4N15', '4N21');
* where type=E & experience=4 ; *&=AND ;
run;
```

9. Using DIF and LAG functions

DIF function creates a new variable (D) by taking the difference of observations of another variable (X) (See SAS manual). Syntax, DIF<n>(argument); *<-- n specifies number of LAGS

```
data two;
input X @@;
Z=lag(x);
D=dif(x);
datalines;
```

```
2 6 4 7;
proc print data=two;
run;

Results of the PROC PRINT step follow:
X Z D
1 . .
2 1 1
6 2 4
4 6 - 2
```

10. Converting a variable into Lowcase or Uppercase

```
data a; set a;
x='INTRODUCTION';
y=lowcase(x);
put y;
run;
```

11. How to put 3 variables of a genotype given in different columns into one column

fam	tree	ht	dbh	vol
1	1	12	4.5	125
1	2	10	3.9	109

```
data a;
set b(rename=(ht=y))
b(rename=(dbh=y))
b(rename=(vol=y));
run;
Result:
```

fam	tree	y
1	1	12
1	1	4.5
1	1	125
1	2	10
1	2	3.9
1	2	109

12. Creating an OUTPUT Text data file (ASCII) from SAS file

```
* The first part creates a file putting Column HEADINGS
only;
data _null_ ;
file 'c:\myfolder\data.txt' ;
put 'factor var1 var2 var3' ; run;

* 2nd part puts data in columns, appends this file to the
first;
data _null_ ;
set mydata;
file 'c:\myfolder\data.txt' mod ;
put factor 1-8 (var1-var2) (7.1) ; run;

* PC: the 'mod' statement appends the second file to the
first.
Delete the output file (data.txt) before running the code
again;
```

13. Transposing statistics

```
data a ;
input chain $ stat $ g1 g2 g3 ;
datalines;
378oneI1 MEAN 0.5130 0.1976 0.7106
378oneI1 MEDIAN 0.4675 0.1703 0.6712
378oneI1 STD 0.3235 0.1339 0.3634
378oneI2 MEAN 0.5257 0.1977 0.7233
378oneI2 MEDIAN 0.4810 0.1707 0.6818
378oneI2 STD 0.3267 0.1309 0.3699
;
proc sort data=a ; by chain ; run;

* If you have Gruoping (here Chain) use BY statement. It is
not transposed.
the ID statement names the variable whose values become new
variable names
VAR variables are the transposed values;
proc transpose data=a out=b;
by chain;
var g1 g2 g3;
id stat;
run;
```

```
data c; set b;
meanstd=input(compress(mean||'±'||std), $12.) ;
run;

proc print data=c noobs; run;
```

chain	_NAME_	MEAN	MEDIAN	STD	meanstd
378onel1	g1	0.5130	0.4675	0.3235	0.513±0.3235
378onel1	g2	0.1976	0.1703	0.1339	0.1976±0.133
378onel1	g3	0.7106	0.6712	0.3634	0.7106±0.363
378onel2	g1	0.5257	0.4810	0.3267	0.5257±0.326
378onel2	g2	0.1977	0.1707	0.1309	0.1977±0.130
378onel2	g3	0.7233	0.6818	0.3699	0.7233±0.369

14. Grouping observations with IF/THEN ELSE statements

```
* grouping variables or creating a new variable using
IF/THEN ELSE statement;

data a;
input dens percent @@;
datalines;
0.453 0.42  0.470 0.46  0.396 0.39  0.430 0.46
;
run;

data a; set a ;
if dens>=0.450 and percent>=0.40 then type='T';
else type='J' ;
run;

Result:

dens   percent type
0.453    0.42    T
0.470    0.46    T
0.396    0.39    J
0.430    0.46    J
```

15. Using * or ? WILDCARDS to read many external files into one file

The * and ? wildcards can be used **in** *either the external file name* or *file extension* for matching input file names. Use **?** to match a single character and the ***** to match one or more characters. The below code combines all the data sets starting with 'wild' in the 'myfolder'.

Example1:

```
filename wild 'd:\myfolder\wild*.dat';
data a;
infile wild;
input var1 var2 var3;
run;
```

'wild2?4.dat' links all the data sets 'wild2**0**4.dat', 'wild2**5**4.dat', 'wild2**A**4.dat', but not 'wild**1**04.dat'

Example2: This code combines all the data sets in the 'myfolder' subdirectory regardles of their extention.

```
data a;
infile 'myfolder/*.*' dlm='09'x dsd missover firstobs=2;
input station week time tmax tmin prec snow;
run;
```

Example3: This code combines all the data sets having .txt extention in the 'myfolder' subdirectory

```
data a;
infile 'myfolder/*.txt' dlm='09'x dsd missover firstobs=2;
input station week time tmax tmin prec snow;
run;
```

16. Adding overall mean to a data set

```
data a;
input x @@;
cards;
5 3 4 2 4 ;

proc means data=a noprint;
output out=sumdata(keep=xmean) mean=xmean;

data b;
if _n_=1 then set sumdata;
set a;
```