Paper 203-27

Transforming Single-Record Spreadsheet Data into Multiple Observations

Glenda Garner, Wake Forest University

ABSTRACT

Generating SAS® datasets from ASCII files is a simple task. Generating datasets from spreadsheet style data is also easily accomplished in SAS. However, the task becomes much more difficult when dealing with a mixed format of data variables with a varying number of input lines per record. One such example was the conversion process of biomechanical gait data into SAS datasets in the ADAPT study(Arthritis, Diet, and Activity Promotion Trial).
This poster will address implementing advanced features of the INPUTstatement.

RAW DATA EXAMPLE

TABLE 1

60030r1w1.xls		
Avg Step Width (cm)	10.81	
R_Velocity	102.007	
R_Stride_Len	122.392	
R_Cadence	98.63	
L_Velocity		
L_Stride_Len		
L_Cadence		
R_Support_Time	64.384	
L_Support_Time		
R_Non_Support	35.616	
L_Non_Support		
R_Step_Len	57.84	
L_Step_Len	64.552	
R_Dbl_Support	16.438	
L_Dbl_Support		
RHS 58	131	
LHS 94		
RTO 105	5	
LTO 70		
RHS FP 59		
LHS FP		
R_HIP Rot ANG	R_HIP Abd ANG	R_HIP Flex ANG
6.608	2.316	62.296
6.748	2.578	60.988

Table 1 shows a portion of the Biomechanical gait data for the ADAPT study. Each participant's data was contained in a separate spreadsheet. Within each spreadsheet, each record consisted of 23 rows of temporal/spatial data on the participant. The next group of rows consist of 87 columns of data with each row representing a time period in the gait cycle. Processing included exporting each spreadsheet from Excel into a CVS file (ASCII comma- delimited format) and then concatenating all the records into one file.

SAMPLE CODE

INPUT Line Features Used:

- Line pointer control, #, moves the pointer to the line number specified.
- Column pointer control, @, moves the pointer to the column specified.
- Line hold specifier, trailing @@, keeps the pointer on the current input line.
- Three input statements are executed:
  1. The first INPUT statement reads the first 23 lines
  2. The second INPUT statement tests for end of record
  3. The third INPUT statement is within the DO UNTIL loop, which creates an output line for each iteration. The loop is executed until end of record or end of file is reached.

Infile Options Used:

- lrecl the logical record length must be given or a default of 80 is used.
- dlm the delimiting character is the comma

TABLE 2

```
libname x '/home/pepper/adapt/gait/datasets/baseline';
filename raw '/home/pepper/adapt/gait/rawdata/baseline/gait.out';
options ls=80;

data one;
length id $ 11;
quit='n';
obnumber=0;
infile raw lrecl=600 dlm="," ;

input  id $       /* The first INPUT statement      */
#2 @20 avg_step
#3 @11 r_vel
#4 @13 r_slen
#5 @10 r_cad
#6 @11 l_vel
#7 @13 l_slen
#8 @11 l_cad
#9 @15 r_time
#10 @15 l_time
#11 @14 r_nsup
#12 @14 l_nsup
#13 @11 r_step
#14 @11 l_step
#15 @14 r_dbl
#16 @14 l_dbl
#17 @4 rhs @7 rhs2
#18 @4 lhs @7 lhs2
#19 @4 rto @7 rto2
#20 @4 lto @7 lto2
#21 @7 rhs_fp1
```

```
#22 @7 lhs_fp
#23  headers $ ;

do until (quit='y');
 input var1 @@;            /*The second  INPUT statement    */

if var1 ne . then do;
    input var2-var87;         /*The third INPUT statement       */
    obnumber = obnumber + 1;
    end;
else
   quit='y';
  output;
end;
```

Table 2 shows the code used in making each of these rows a record with the temporal/spatial data included in the observation.

SAMPLE OUTPUT DATA

TABLE 3

Obs	ID	AVG_STEP	R_VEL	R_SLEN
1	60030r1w1.x	10.81	102.007	122.392
2	60030r1w1.x	10.81	102.007	122.392
3	60030r1w1.x	10.81	102.007	122.392
4	60030r1w1.x	10.81	102.007	122.392

Obs	RTO2	LTO	Var1	Var2	Var3	Var4
1	5	70	6.608	2.316	62.296	-21.889
2	5	70	6.748	2.578	60.988	-21.312
3	5	70	6.556	2.632	59.652	-20.490
4	5	70	5.953	2.371	58.303	-19.358

Table 3 shows a portion of the first four records of the output. The temporal/spatial data is repeated for each observation. All variables are kept in the final dataset. The ID variable identifies the participant ID, the visit number, more affected side, and the trial number. These data will be analyzed to determine any significant differences or changes between or within the intervention groups.

CONCLUSION

This paper demonstrates how to write a SAS program to read ASCII data with a non-fixed format and a varying number of lines per record. Additionally shown is how to create output with multiple observations from one single record.

References
SAS Institute Inc(1990) SAS Language, Cary, NC: SAS Institute Inc, 420-421 pp.