Chapter 21 CREATING MULTIPLE OBSERVATIONS

FROM A SINGLE RECORD

Sometimes multiple records appear on the same line. It reduces the size of the entire data file.

READING REPEATING BLOCKS OF DATA

If each line contains values of the same variables for different individuals (called repeating blocks of data), then the records may be read using the **double “at” sign** (@@) as a **line-hold specifier**. It holds the record across multiple iterations of the DATA step until the end of the record is reached. The syntax is

input *variable1 variable2* ... @@;

Example. The program below reads repeated blocks of data containing student ID, and scores on a quiz, homework, exam and the final.

data scores;

input student\_id$ quiz hw exam final @@;

cards;

4533 45 32 91 112 614 50 35 98 108 1185 30 23 76 92

8018 94 98 100 119 2669 34 67 74 92 177 54 84 88 88

;

proc print noobs;

run;

The output is

student\_

id quiz hw exam final

4533 45 32 91 112

614 50 35 98 108

1185 30 23 76 92

8018 94 98 100 119

2669 34 67 74 92

177 54 84 88 88

READING THE SAME NUMBER OF REPEATING FIELDS

If each line contains an individual’s ID followed by some number of repeating fields, and the number of repeating fields is the same for all individuals, then the records may be read using the **single “at” sign** (@) as a **line-hold specifier**. It holds the input record for the next INPUT statement in the DATA step. The syntax is

input *variable* @;

Example. A student’s ID number and scores on four quizzes are stored on one line. The data are

ID Quiz1 Quiz2 Quiz3 Quiz4

1534 90 87 78 92

The program below pairs the ID number with the score on each quiz and creates four lines of data.

data quizscores;

input ID$ @;

do quiz=1 to 4;

input score @;

output;

end;

cards;

1534 90 87 78 92

;

proc print noobs;

run;

The output is

ID quiz score

1534 1 90

1534 2 87

1534 3 78

1534 4 92

READING A VARYING NUMBER OF REPEATING FIELDS

If each line contains an individual’s ID followed by some number of repeating fields, and the number of repeating fields varies across individuals, then the records may be read using the **MISSOVER option** in the INFILE statement. The reason behind using this option is that records with smaller number of fields are considered containing missing values. The **DO WHILE loop** may be used to pair individual’s ID with each of the fields and output them on a separate line.

Example. A student’s ID number and scores on several quizzes are stored on one line. The data are

ID Quiz1 Quiz2 Quiz3 Quiz4 Quiz5 Quiz6

1534 90 87 78 92

3008 54 58 63 73 78 77

The program below pairs the ID number with the score on each quiz and creates ten lines of data.

data quizscores;

infile 'F:\quiz\_scores.dat' missover;

input ID$ score @;

quiz=0;

do while (score ne .);

quiz+1;

output;

input score @;

end;

run;

proc print noobs;

run;

The output is

ID score quiz

1534 90 1

1534 87 2

1534 78 3

1534 92 4

3008 54 1

3008 58 2

3008 63 3

3008 73 4

3008 78 5

3008 77 6

Chapter 22 READING HIERARCHICAL FILES

Data file is called **hierarchical** if it consists of a **header record** and one or more **detail records**. The header record is common to all detail records.

CREATING ONE OBSERVATION PER DETAIL RECORD

The objective is to build a SAS data set that contains one observation per detail record and the corresponding header record is stored as part of the observation.

Example. The following data contain course number, names and status (senior/graduate) of several students in this course.

STAT 475

John B. Graduate

Dan C. Graduate

Paul C. Senior

Michelle G. Graduate

Mat N. Graduate

STAT 590

Jeff L. Graduate

Martha N. Graduate

Kelvin N. Graduate

Alyssa A. Senior

Francisco R. Graduate

The following program reads the data file that contains these data along with the type of record ('C'='course' is the header record, and 'S'='student' is the detail record). It creates a data set where each observation contains course number, student’s name and status.

data students (drop=type);

infile 'F:\students.dat';

retain Course;

input type $1. @;

if type='C' then

input @3 Course $8.;

if type='S';

input @3 Name $13. @16 Status $8.;

run;

proc print noobs;

run;

The output is

Course Name Status

STAT 475 John B. Graduate

STAT 475 Dan C. Graduate

STAT 475 Paul C. Senior

STAT 475 Michelle G. Graduate

STAT 475 Mat N. Graduate

STAT 590 Jeff L. Graduate

STAT 590 Martha N. Graduate

STAT 590 Kelvin N. Graduate

STAT 590 Alyssa A. Senior

STAT 590 Francisco R. Graduate

CREATING ONE OBSERVATION PER HEADER RECORD

The objective is to build a SAS data set that contains one observation per header record and combining information from detail records for that header.

Example. The following data contain patient ID, and the length of hospital stay (in days) during a one-year period for each of the patients.

1202

14

22

1832

2

1596

13

6

8

The following program reads the data file that contains these data along with the type of record ('P'='patient' is the header record, and 'H'='hospital' is the detail record). It creates a data set where each observation contains patient ID and a total number of days of hospital stay.

data hospital (drop=type stay\_days);

infile 'F:\hospital\_stay.dat' end=last;

retain patientID;

input type $1. @;

if type='P' then do;

if \_n\_>1 then output;

totalstay=0;

input @3 patientID $4.;

end;

else if type='H' then do;

input @3 stay\_days 2.;

totalstay+stay\_days;

end;

if last then output;

run;

proc print noobs;

run;

The output is

patient

ID totalstay

1202 36

1832 2

1596 27