Chapter 13 COMBINING SAS DATA SETS

In SAS programming, sometimes it is needed to combine observations from two or more data sets into a new data set. Four methods of combining data sets are commonly used.

* One-to-one reading
* Concatenating
* Interleaving
* Match-merging

ONE-TO-ONE READING

The syntax is

data *dataset\_name*;

set *dataset1*;

set *dataset2*;

run;

The rules of one-to-one reading are as follows:

* The new data set contains **all the variables** from all the input data sets. If some variables have the same name, the values read from the last data set overwrite the earlier read values.
* The number of observations in the new data set is the number of observations in the **smallest** input data set.

Consider data set A and data set B

ID VarB

1 B1

2 B2

3 B3

ID VarA

1 A1

2 A2

3 A3

4 A4

The following SAS program combines data sets A and B with one-to-one reading.

data A;

input id$ VarA$;

cards;

1 A1

2 A2

3 A3

4 A4

;

data B;

input id$ VarB$;

cards;

1 B1

2 B2

3 B3

;

data one\_to\_one;

set A;

set B;

run;

proc print data=one\_to\_one noobs;

run;

The result is

id VarA VarB

1 A1 B1

2 A2 B2

3 A3 B3

Example. Two data sets, one containing demographic variables and the other containing quiz scores for the same individuals, are combined one-to-one. Note that in order to combine the data sets correctly, the observations must be listed by ID in the same way in both data sets.

data demogr;

input id$ age race$;

cards;

002 18 hisp

034 22 white

010 20 hisp

011 15 hisp

017 21 white

;

data quiz;

input id$ score;

cards;

002 64

034 76

010 92

011 88

;

data one\_to\_one;

set demogr;

set quiz;

run;

proc print noobs;

run;

The result is

id age race score

002 18 hisp 64

034 22 white 76

010 20 hisp 92

011 15 hisp 88

CONCATENATING

The syntax is

data *dataset\_name*;

set *dataset1* *dataset2*;

run;

**Concatenating** appends the observations from dataset2 to those from dataset1. The following SAS program concatenates data sets A and B.

data concat;

set A B;

run;

proc print noobs;

run;

The result is

id VarA VarB

1 A1

2 A2

3 A3

4 A4

1 B1

2 B2

3 B3

Example. Two data sets contain demographic variables on two different groups of individuals. The data sets may be concatenated to combine the information. Note that in order to combine the data sets correctly, the variables in both data sets must have the same names.

data demogr1;

input id$ age race$;

cards;

002 18 hisp

034 22 white

010 20 hisp

011 15 hisp

;

data demogr2;

input id$ age race$;

cards;

012 28 white

017 21 white

088 18 hisp

110 35 hisp

;

data concat;

set demogr1 demogr2;

run;

proc print noobs;

run;

The result is

id age race

002 18 hisp

034 22 white

010 20 hisp

011 15 hisp

012 28 white

017 21 white

088 18 hisp

110 35 hisp

INTERLEAVING

The syntax is

data *dataset\_name*;

set *dataset1* *dataset2*;

by *variablename(s)*;

run;

**Interleaving** intersperses observations from dataset1 and dataset2, based on values of one or more common variables. **Note that the data sets must be ordered in ascending order with respect to these common variables.**

The following SAS program interleaves data sets A and B by id.

data interleaved;

set A B;

by id;

run;

proc print noobs;

run;

The result is

id VarA VarB

1 A1

1 B1

2 A2

2 B2

3 A3

3 B3

4 A4

Example. If data sets demogr1 and demogr2 are first sorted by id, then they may be combined by interleaving. The following code produces the combined data set.

proc sort data=demogr1;

by id;

run;

proc sort data=demogr2;

by id;

run;

data interleaved;

set demogr1 demogr2;

by id;

run;

proc print noobs;

run;

The resulting data set is

id age race

002 18 hisp

010 20 hisp

011 15 hisp

012 28 white

017 21 white

034 22 white

088 18 hisp

110 35 hisp

MATCH-MERGING

The syntax is

data *dataset\_name*;

merge *dataset1* *dataset2*;

by *variablename(s)*;

run;

**Match-merging** combines observations from dataset1 and dataset2, based on values of one or more common variables. **Note that the data sets must be ordered in ascending or descending order with respect to these common variables.**

The following SAS program merges data sets A and B by id.

data merged;

merge A B;

by id;

run;

proc print noobs;

run;

The result is

id VarA VarB

1 A1 B1

2 A2 B2

3 A3 B3

4 A4

Example. If data sets demogr and quiz are first sorted by id, then they may be combined by match-merging. The following code produces the merged data set.

proc sort data=demogr;

by id;

run;

proc sort data=quiz;

by id;

run;

data merged;

merge demogr quiz;

by id;

run;

proc print noobs;

run;

The merged data set is

id age race score

002 18 hisp 64

010 20 hisp 92

011 15 hisp 88

017 21 white .

034 22 white 76

* If merging is done by values of some variable which are ordered in **descending**

order, then the option **DESCENDING** must be specified in after the MERGE

statement. The following SAS program merges data sets A and B by id in descending order.

proc sort data=A;

by descending id;

run;

proc sort data=B;

by descending id;

run;

data merged;

merge A B;

by descending id;

run;

proc print noobs;

run;

The result is

id VarA VarB

4 A4

3 A3 B3

2 A2 B2

1 A1 B1

Example. If data sets demo and quiz are sorted by id in **descending order**, then they may be combined by match-merging. The following code produces the merged data set.

proc sort data=demogr;

by descending id;

run;

proc sort data=quiz;

by descending id;

run;

data merged;

merge demogr quiz;

by descending id;

run;

proc print noobs;

run;

The merged data set is

id age race score

034 22 white 76

017 21 white .

011 15 hisp 88

010 20 hisp 92

002 18 hisp 64

RENAMING VARIABLES

If several data sets have **same-named** **variables**, the variables must be renamed before merging the data sets. Otherwise, the variables become overwritten by subsequent values as files are merged.

Consider data set A and data set C

ID VarA

1 C1

2 C2

3 C3

ID VarA

1 A1

2 A2

3 A3

4 A4

Both data sets contain variable VarA. The following program merges the data sets.

data A;

input id$ VarA$;

cards;

1 A1

2 A2

3 A3

4 A4

;

data C;

input id$ VarA$;

cards;

1 C1

2 C2

3 C3

;

data merged;

merge A C;

by id;

run;

proc print noobs;

run;

The merged data set is

id VarA

1 C1

2 C2

3 C3

4 A4

Note that the values of VarA in data set A are overwritten by all available values in data set B.

To prevent overwriting, use the option **RENAME=** in the MERGE statement.

The syntax is

data *dataset\_name*;

merge *dataset1*(rename=(*old\_variable\_name*=*new\_variable\_name1*))

*dataset2*(rename=(*old\_variable\_name*=*new\_variable\_name2*));

by *variablename(s)*;

run;

* Note that it is sufficient to rename the same-named variable(s) only in one data set, not in both.

The following program renames VarA and merges data sets A and C correctly.

data merged;

merge A C(rename=(VarA=VarC));

by id;

run;

proc print noobs;

run;

The merged data set is

id VarA VarC

1 A1 C1

2 A2 C2

3 A3 C3

4 A4

Example. Two data sets contain scores on quizzes 1 and 2 for the same individuals. To merge the files, it is necessary to rename the same-named variable score. The following code renames the variable in both data sets and merges the files properly.

data quiz1;

input id$ score;

cards;

002 64

034 76

010 92

011 88

;

data quiz2;

input id$ score;

cards;

002 84

034 82

010 96

011 93

;

proc sort data=quiz1;

by id;

run;

proc sort data=quiz2;

by id;

run;

data merged;

merge quiz1 (rename=(score=score1))

quiz2 (rename=(score=score2));

by id;

run;

proc print noobs;

run;

The merged data set is

id score1 score2

002 64 84

010 92 96

011 88 93

034 76 82

EXCLUDING UNMATCHED OBSERVATIONS

When merging data sets, if it is desired to select only observations that match for two or more input data sets, the option **IN=** and the **subsetting IF statement** may be used. The syntax is

data *dataset\_name*;

merge *dataset1*(in=*variable\_name1*)

*dataset2*(in=*variable\_name2*);

by *variablename(s)*;

if *variable\_name1*=1 and *variable\_name2*=1;

run;

* Here variable\_name1 and variable\_name2 are **temporary** variables.
* They assume value 1 if the corresponding data set contributed to the current observation, and 0, otherwise.
* The subsetting IF statement instructs SAS to use only those observations that are present in both data sets.
* Alternatively, the IF statement may be written as

if *variable\_name1* and *variable\_name2*;

This expression checks for a value that is **neither missing nor 0**, which in this case is tantamount to being equal to 1.

Consider data set A and data set B

ID VarB

1 B1

2 B2

3 B3

ID VarA

1 A1

2 A2

3 A3

4 A4

The fourth observation in data set A is unmatched in data set B. The code below merges the data sets and removes the unmatched observation.

data merged;

merge A(in=checkA) B(in=checkB);

by id;

if checkA and checkB;

run;

proc print noobs;

run;

The merged data set is

id VarA VarB

1 A1 B1

2 A2 B2

3 A3 B3

Example. The data set demogr has one individual with id=017, for whom the score was not recorded in the data set quiz. The following code produces the merged data set that excludes the unmatched individual.

data demogr;

input id$ age race$;

cards;

002 18 hisp

034 22 white

010 20 hisp

011 15 hisp

017 21 white

;

data quiz;

input id$ score;

cards;

002 64

034 76

010 92

011 88

;

proc sort data=demogr;

by id;

run;

proc sort data=quiz;

by id;

run;

data merged;

merge demogr(in=indemogr) quiz(in=inquiz);

by id;

if indemogr and inquiz;

run;

proc print noobs;

run;

The merged data set is

id age race score

002 18 hisp 64

010 20 hisp 92

011 15 hisp 88

034 22 white 76

SELECTING VARIABLES WITH DROP OPTION

To drop variables, use the option **DROP=** in the DATA step or the MERGE statement. The syntax is

data *dataset\_name*(drop*=var\_name1 var\_name2 …*);

merge *dataset1*(drop=*variable1 variable2 …*)

*dataset2*(drop=*variable\_name1 variable\_name2 …*);

by *variablename(s)*;

run;

* Here *variable1, variable2*, etc. and *variable\_name1, variable\_name2,* etc. are dropped **before** being processed by the MERGE statement.
* The variables *var\_name1, var\_name2*, etc. are dropped **after** the files are merged. This variable may be in use during the merging. For example, it may be one of the BY statement variables.

Example. The following SAS program merges data sets demogr and quiz, dropping the variable race before merging and dropping the variable id after merging.

data demogr;

input id$ age race$;

cards;

002 18 hisp

034 22 white

010 20 hisp

011 15 hisp

017 21 white

;

data quiz;

input id$ score;

cards;

002 64

034 76

010 92

011 88

;

proc sort data=demogr;

by id;

run;

proc sort data=quiz;

by id;

run;

data merged (drop=id);

merge demogr (drop=race) quiz;

by id;

run;

proc print noobs;

run;

The merged data set is

age score

18 64

20 92

15 88

21 .

22 76

SELECTING VARIABLES WITH KEEP OPTION

If input data sets have a large number of variables, but merged data set should contain only several of them, it is easier to use the option **KEEP=** instead of DROP=.

Example. Data set grades contains scores on quizzes 1 through 5, exams 1 through 3, and the final. The program below merges data sets demogr and grades and keeps only variables id, age and final.

data demogr;

input id$ age race$;

cards;

002 18 hisp

034 22 white

010 20 hisp

011 15 hisp

017 21 white

;

data grades;

input id$ quiz1-quiz5 exam1-exam3 final;

cards;

002 64 54 70 67 73 75 75 70 80

034 76 78 80 88 81 82 84 83 88

010 92 91 90 88 94 90 91 91 92

011 88 84 80 82 78 64 78 82 81

;

proc sort data=demogr;

by id;

run;

proc sort data=grades;

by id;

run;

data merged;

merge demogr (drop=race) grades (keep=id final);

by id;

run;

proc print noobs;

run;

The output is

id age final

002 18 80

010 20 92

011 15 81

017 21 .

034 22 88