3. Categorizing and Cross-Classifying Data in CREATE

3.1 Overview

The issues of categorizing and cross-classifying data in VPLX are closely linked but nonetheless distinct. This chapter combines the topics but presents them in separate sections.

Categorical data are commonplace in surveys and many other statistical applications. Even variables with underlying continuous distributions, such as age, may be collected and analyzed categorically. Estimated totals and percentages are among basic statistical products associated with categorical data.

Discrete variables may also be used to cross-classify other variables; in this role they are termed *class variables* in VPLX. Class variables may cross-classify both categorical variables, such as labor force status, and real variables, such as earnings. Any survey variable that can be analyzed categorically can be made into a class variable instead. It is also possible to form both categorical and class variables from the same information. For example, the percentage distribution based on a categorical variable for age and cross-classifications of other variables based on a class variable for age can be obtained simultaneously.

Section 3.2 covers the CATEGORICAL or CAT statement, which identifies how the levels of a categorical variable are to be identified from a real variable. These statements appear in the categorization section of the standard order for the CREATE step shown in Exhibit 1.1.

Section 3.3 introduces both CLASS and BLOCK statements, which work together to specify cross-classifications. CLASS statements, which appear in the categorization section along with any CATEGORICAL statements, define individual class variables that may be used as cross-classifiers. The BLOCK statements that follow in the output section specify how other variables are to be cross-classified by the CLASS variables. Section 3.4 describes LABEL, which may be included in the output section. Section 3.5 addresses a more specialized form of cross-classification suitable for large applications through BY variables.

3.2 CATEGORICAL (CAT) Statements

The CATEGORICAL statement converts a real variable into a categorical variable. CAT may abbreviate CATEGORICAL. Range specifications indicate what values of the real variable comprise each level of the resulting categorical variable. Labels of up to 24 characters for each level may follow. For example,

```
cat sex (1/2) 'Male' 'Female' ;
```

This example illustrates a simple case in which a single variable is converted from a real variable into categorical. Following the variable name, one or more sets of ranges follow, separated by "/," and bounded by opening and closing parentheses. In this case, real values of 1 are converted into the first level of the categorical variable, and 2 into the second. If the real variable has some other value not covered by the specified ranges, such as 0 or 9 in this instance, then the observation is excluded from the resulting categorical variable.

Section 2.5.3 presented rules for range specifications, which are identical in IF, CAT, and CLASS statements. Generally, "res" for residual is not used with IF but instead with CAT and CLASS. In this instance,

```
cat sex (1/2/res) 'Male' 'Female' 'DK';
```

the use of "res" places all other real values into a third level. The "low" and "high" specifications are also useful with CAT and CLASS statements.

```
cat income (low- 0/ 1 - 19999/20000-39999/40000-59999/
        60000-79999/80000 - 99999 /100000 - high)
         'None or loss' '$1 - $19,9999'
         '$20,000 - $39,999' '$40,000 - $59,999'
         '$60,000 - $79,999' '$80,000 - $99,999'
         '$100,000 or more';
```

Note that in the previous example each level is matched to an accompanying label. If no labels are given, then the labels are set to the first 24 characters of the ranges. For example,

```
cat age (0-17/ 18-29/30-44/45-64/65-high) ;
```

is equivalent to,

```
cat age (0-17/18-29/30-44/45-64/65-high) '0-17' '18-29' '30-44'
         '45-64' '65-high';
```

Note also that the labels enclosed in apostrophes are separated by one or more blanks. Commas may also be used to separate labels. If two apostrophes are adjacent, then they are translated into a single apostrophe within the label,

```
cat  q103 (1/2/3,4/res) 'Yes', 'No', 'Don''t know or refused',
     'Missing/no answer';
```

CAT statements may be used to categorize several variables simultaneously. The same range specifications and labels are then applied to each.

```
cat  q103 - q117 q121 q123 (1/2/3,4/res) 'Yes', 'No',
   'Don''t know or refused','Missing/no answer';
```

The extended SIPP example from Exhibit 1.5 included the following,

```
 if pp_intvw1 .in. {1,2} .and. age1 >= 15 then;
    if grd_cmpl1 == 1 then;                          ! completed grade
      hicmpgrd1 = higrade1 ;
    else if grd_cmpl1 == 2 then;                     ! grade not complete
       if higrade1 == 0 .or. higrade1 == 21 .or.     ! do not adjust
          higrade1 == 25 then;                       ! 21 (some college)
         hicmpgrd1 = higrade1;                       ! 25 (some grad sch)
       else;
         hicmpgrd1 = higrade1 - 1;                   ! adjust other grades
       end if;
    end if;
 end if;

...similar code for hicmpgrd2, hicmpgrd3, and hicmpgrd4 for waves 2-4

 cat hicmpgrd1 - hicmpgrd4 (0-11/12/21-23/24/25-26)
             'Less than HS' 'HS graduate' 'Some college'
             'College grad' 'Some post-grad';

 ...

display

...

list hicmpgrd1 - hicmpgrd4

...

                                                  Estimate      Standard error
Completed ed., intvw 1  :  PERCENTS
   Less than HS                                    25.8180              .5240
   HS graduate                                     34.0098              .4617
   Some college                                    22.2100              .3438
   College grad                                     9.1430              .2843
   Some post-grad                                   8.8193              .2695

...
```

Exhibit 3.1 An extract from i1-3.lis. A variable corresponding to highest completed grade is first constructed within the recode section of the CREATE step. The real variables are then converted to categorical with a single CAT statement. The DISPLAY gives percentages for the resulting variable. A display of estimated totals is also possible, but was not included in the example.

The example illustrates an important point about CAT statements: the syntax specifies ranges for a single variable at a time to determine level assignment. When levels defined by more complex relationships involving two or more variables are required, a variable summarizing these relationships must be constructed within the recode section first.

All of the previous examples convert real variables into categorical variables. At the point in the program where the CAT statement appears, VPLX converts the real variable into the resulting categorical level, and the original real value is lost. In many applications, however, the original value is need for other purposes. There are two programming strategies to preserve the original value. The first is to define as many different versions of the real variable as necessary in the recode section and then to change each of them into categorical variables. For example, if two different age groupings were of interest,

```
{age5 age10} = age;
cat age5 (0-4/5-9/10-14/15-19/20-24/25-29/30-34/35-39/40-44/
          45-49/50-54/55-59/60-64/65-69/70-74/75-high) ;
cat age10 (0-9/10-19/20-29/30-39/40-49/50-59/60-69/70-high) ;
```

produces two different categorical variables without changing the status of age as a real variable. Alternatively,

```
cat age into age5 (0-4/5-9/10-14/15-19/20-24/25-29/30-34/35-39/40-44/
             45-49/50-54/55-59/60-64/65-69/70-74/75-high) ;
cat age into age10 (0-9/10-19/20-29/30-39/40-49/50-59/60-69/70-high) ;
```

achieves the same effect. When INTO appears in the command, it is not treated as a variable name but instructs VPLX to create a new categorical variable while leaving the real variable unchanged. As a consequence of this syntax, INTO cannot be used as a variable name (*1*).

Finally, the INTO feature may be combined with the ability to categorize several variables at once. The syntax takes the form:

```
CAT  vlist1 INTO vlist2 (range1 / range2 ...) ['label1 ' ['label2'[...]]] ;
```

where *vlist1* and *vlist2* must have the same length, that is, the same number of variables. New variables appearing in *vlist2* become defined by this statement.

The following example is based on the same simple data set as Exhibit 1.3.

```
create  in = i1-1.dat  out = vplx1.vpl ;  ! Beginning of CREATE step


  input  rooms persons  weight cluster    ! input statement to read
          stratum /format (5F2.0) ;       ! data  ("input section")
```

```
  cat rooms into rooms1 (4-5/6-high)      ! Creating a 2-level variable
      '4-5 rooms' '6 or more rooms';      ! from rooms, with labels

  cat rooms into rooms2 (4-5/6-7)         ! A 2-level variable, where 8+
      '4-5 rooms' '6-7 rooms';            ! are dropped

  cat rooms(4-5/6-7/8);                   ! This converts rooms into a
                                          ! 3-level categorical variable,
                                          ! with default labels for levels.

  block rooms1 rooms2 rooms ;             ! Block statement to keep the
                                          ! 3 categorical variables

   ... diagnostic information from VPLX omitted

display                                   ! Beginning of DISPLAY step

list rooms1 rooms2 rooms                  ! specification of
     total (rooms1 rooms2 rooms)          ! display

                                                 Estimate      Standard error

rooms1                  :  PERCENTS
   4-5 rooms                                      50.0000            16.6667
   6 or more rooms                                50.0000            16.6667

rooms2                  :  PERCENTS
   4-5 rooms                                      75.0000            25.0000
   6-7 rooms                                      25.0000            25.0000

rooms                   :  PERCENTS
   4-5                                            50.0000            16.6667
   6-7                                            16.6667            16.6667
   8                                              33.3333              .0000

rooms1                  :  TOTAL
   4-5 rooms                                       3.0000             1.0000
   6 or more rooms                                 3.0000             1.0000

rooms2                  :  TOTAL
   4-5 rooms                                       3.0000             1.0000
   6-7 rooms                                       1.0000             1.0000

rooms                   :  TOTAL
   4-5                                             3.0000             1.0000
   6-7                                             1.0000             1.0000
   8                                               2.0000              .0000
```

Exhibit 3.2 An extract from i3-1.lis. Three categorical variables are created from rooms. The second omits 2 cases. The percent distribution shown by display is based on the cases falling into one of the defined levels, so that the estimated percent with 4-5 rooms for the second variable disagrees with the first and third.

3.3 CLASS and BLOCK

CLASS and BLOCK statements are designed to work together. The CLASS statement is syntactically identical to the CAT statement, except to begin with CLASS. As a simple example, analogous to the first in section 3.2,

```
class sex (1/2) 'Male' 'Female' ;
```

defines sex as a potential cross-classifier for other variables. All of the other syntactic options for the CATEGORICAL statement, such as the definition of multiple class variables in a single statement, are available to CLASS as well.

Frequently, a variable is called upon for both categorical and classification purposes. In these cases, separate categorical and class variables should be created. For example,

```
cat sex into sex1 (1/2) 'Male' 'Female' ;
class sex (1/2) 'Male' 'Female' ;

block sex1 ;
block unemployed / class sex ;

display
list percent (sex1)
list unemployed / class sex
```

Once a class variable is defined by a CLASS statement, it becomes available for use as a cross-classifier by one or more BLOCK statements in the output section. In general, a BLOCK statement consists of BLOCK and one or more of the following:

- a list of variables associated with the block, excluding class variables,
- a /CLASS specification of associated class variables,
- one or more /SELECT specifications giving conditions that must be satisfied to include an observation in the block. If more than one /SELECT specification is present, observations must satisfy each of the /SELECT conditions for inclusion.

The following BLOCK statement from Exhibit 1.5 illustrates all three aspects,

```
block c_esr1 / class sex * race * ethnicity
      / select if c_pp_mis1 .in. {1}
      / select if c_age1 .in. {15-high};
```

This block cross-classifies C_ESR1 by sex, race, and ethnicity, a 2 (sex) by 4 (race) by 2 (ethnicity) cross-classification. The subsequent two /SELECT specifications each impose conditions on inclusion of the observation. Note that the /SELECT specifications use the ".in. { }" syntax appearing in the IF statement. No other logical expressions are allowed, so for more complex selection criteria, a single variable must be defined in the recode section of the CREATE step for use with the ".in. { }" syntax of /SELECT (*2*).

The BLOCK statements determine the contents of the output VPLX file. For each block defined in the CREATE step, a weighted count (the sum of weights for observations included in the block) and totals for the associated variables are aggregated, cross-classified by the class variables, and subject to the /SELECT conditions. If no weight variable is present, then all totals are unweighted. The DISPLAY step uses the weighted count for the block to calculate means of real variables. The DISPLAY step can also display the weighted count associated with any block.

Although class variables can be used to cross-classify more than one block, other variables can be associated with at most one block. If equivalent information is required in more than one block, then different variables must be created. The list of associated variables should not include the weight variable, if it is present, or any stratum, cluster, replicate weight or other variables used in defining the replication method.

BLOCK statements exclude observations from the totals for the associated variables in two ways:

- if the conditions of one or more /SELECT conditions are not met, or

- if the observation does not fall into one of the defined levels of one or more of the associated class variables.

It is important to keep both of these conditions in mind in using BLOCK and CLASS statements. Indeed, if a DISPLAY step shows systematically lower totals than expected, the problem may be due to exclusion of cases because one or more class variables does not have a valid level.

If no BLOCK statements appear, then a single block is created by associating all regular variables, cross-classified by any defined CLASS variables. For example, Exhibit 1.4 showed a CREATE step with only an INPUT statement; in this case the variables rooms and persons were associated with a single block.

For simple applications, reliance on the default by omitting BLOCK statements may give satisfactory results unless variables pertain to different universes requiring /SELECT in separate blocks. In more complex applications involving the definition of many variables, however, 20 or more CLASS variables may be required, and the computer resources can be rapidly exhausted by producing the

complete cross-classification of other variables by all class variables. BLOCK statements specify the class variables to use with associated variables and therefore make large applications possible.

To illustrate several of these points, consider two possible applications, based on modifying part of the SIPP example shown in Exhibit 1.5,

```
cat  c_esr1 (1/2/3/4/5/6/7/8) 'Worked all weeks'
       'Job,miss 1+ wk,no layoff', 'Job, some time on layoff',
       'Part job, no layoff/look', 'Part job, w layoff/look',
       'No job, all look/layoff', 'No jb, some look/layoff',
       'No job, no look/layoff';
class sex (1/2) 'Male' 'Female';
class race (1/2/3/4) 'White' 'Black' 'Amer. Indians'
                     'Asian and Pacific Islanders';
class ethnicity (14-20/res) 'Hispanic origin' 'Non-Hispanic';
block c_esr1 / class sex * race * ethnicity;
```

compared to

```
{c_esr1a c_esr1b c_esr1c} = c_esr1;
cat  c_esr1a c_esr1b c_esr1c (1/2/3/4/5/6/7/8) 'Worked all weeks'
       'Job,miss 1+ wk,no layoff', 'Job, some time on layoff',
       'Part job, no layoff/look', 'Part job, w layoff/look',
       'No job, all look/layoff', 'No jb, some look/layoff',
       'No job, no look/layoff';
class sex (1/2) 'Male' 'Female';
class race (1/2/3/4) 'White' 'Black' 'Amer. Indians'
                     'Asian and Pacific Islanders';
class ethnicity (14-20/res) 'Hispanic origin' 'Non-Hispanic';
block c_esr1a / class sex;
block c_exr1b / class race;
block c_exr1c / class ethnicity;
```

Because of the use of res in the class statement for ethnicity, all observations will be assigned one of the two levels. Unexpected values for sex or race, however, will cause the variable to be omitted from the single block in the first version but only the affected block in the second version. For example, if a code of 5 appears for race on some observations, those observations are dropped from the single block in the first case but only dropped from the second of the three blocks in the second case. The first example results in a larger output VPLX file than the second, but allows all possible cross-classifications by the class variables, whereas the second would only permit the display of employment status by one class variable at a time. The first example requires fewer statements and is somewhat easier to program. Thus, small to moderate sized applications such as this (*3*) should tend to use the first programming strategy, which is to use several class variables at once in relatively few BLOCK statements. For large applications, however, it may be necessary to limit the size of the output VPLX file to only the information required for subsequent operations and display by defining more specialized blocks.

Exhibit 3.3 shows an example based on modifying the situation presented in Exhibit 3.2, by forming class variables rather than categorical variables from rooms.

```
/*   The input data are the six records of i1-1.dat:
 5 7 1 1 1
 6 8 1 2 1
 5 2 1 3 2
 4 1 1 4 2
 8 4 1 5 3
 8 2 1 6 3
*/

create  in = i1-1.dat  out = vplx1.vpl ;  ! Beginning of CREATE step


  input  rooms persons  weight cluster    ! input statement to read
          stratum /format (5F2.0) ;       ! data  ("input section")

  {persons1 - persons3} = persons;        ! make 3 copies of persons

  class rooms into rooms1 (4-5/6-high)    ! Creating a 2-level variable
      '4-5 rooms' '6 or more rooms';      ! from rooms, with labels

  class rooms into rooms2 (4-5/6-7)       ! A 2-level variable, where 8+
      '4-5 rooms' '6-7 rooms';            ! are dropped

  class rooms (4-5/6-7/8);                ! This converts rooms into a
                                          ! 3-level class variable,
                                          ! with default labels for levels.

  block persons1 / class rooms1 ;         ! Separate block statements for
  block persons2 / class rooms2 ;         ! each of the three class variables
  block persons3 / class rooms ;          ! to show their different effects


    Stratified jackknife replication assumed

    Size of block   1  =             4

    Size of block   2  =             4

    Size of block   3  =             6

    Total size of tally matrix =     14

**** End of CREATE specification/beginning of execution

    End of primary input file after obs #       6

         3 strata observed on incoming file


display                                   ! Beginning of DISPLAY step

list total (persons1) /class rooms1 /     ! list specification to show
     total (persons2) /class rooms2 /     ! each block in order
     total (persons3) /class rooms
```

```
    rooms1                :   4-5 rooms
                                                  Estimate      Standard error
persons1              :  TOTAL                     10.0000              7.0711

    rooms1                :   6 or more rooms
                                                  Estimate      Standard error
persons1              :  TOTAL                     14.0000              8.2462

    rooms2                :   4-5 rooms
                                                  Estimate      Standard error
persons2              :  TOTAL                     10.0000              7.0711

    rooms2                :   6-7 rooms
                                                  Estimate      Standard error
persons2              :  TOTAL                      8.0000              8.0000

    rooms                 :   4-5
                                                  Estimate      Standard error
persons3              :  TOTAL                     10.0000              7.0711

    rooms                 :   6-7
                                                  Estimate      Standard error
persons3              :  TOTAL                      8.0000              8.0000


    rooms                 :   8
                                                  Estimate      Standard error
persons3              :  TOTAL                      6.0000              2.0000
```

Exhibit 3.3 An extract from i3-2.lis. Parallel to i3-1.lis in Exhibit 3.2, three class variables are created from rooms. The second omits 2 cases. Following the BLOCK statements, the report from the CREATE step provides the size of each block. For example, the first block is of size 4, representing the product of the 2 cells for the weighted number of observations and for persons1 times the 2 levels of the class variable, rooms1. The DISPLAY step shows that 2 observations with 6 people are excluded from the second block, because the class variable rooms2 excluded the two observations with 8 rooms.

Exhibit 1.5 included the following CLASS and BLOCK statements

```
class sex (1/2) 'Male' 'Female';
class race (1/2/3/4) 'White' 'Black' 'Amer. Indians'
                     'Asian and Pacific Islanders';
class ethnicity (14-20/res) 'Hispanic origin' 'Non-Hispanic';
class c_tenure1 - c_tenure12 (2/res) 'Renter' 'Owner or other';
cat  c_esr1 - c_esr12 (1/2/3/4/5/6/7/8) 'Worked all weeks'
       'Job,miss 1+ wk,no layoff'
       'Job, some time on layoff'
       'Part job, no layoff/look'
       'Part job, w layoff/look'
       'No job, all look/layoff'
       'No jb, some look/layoff'
       'No job, no look/layoff';

... label statements omitted

cat hicmpgrd1 - hicmpgrd4 (0-11/12/21-23/24/25-26)
             'Less than HS' 'HS graduate' 'Some college'
             'College grad' 'Some post-grad';

block hicmpgrd1 / select if pp_intvw1 .in. {1}
                / select if age1 .in. {15- high};
block hicmpgrd2 / select if pp_intvw2 .in. {1}
                / select if age5 .in. {15- high};
block hicmpgrd3 / select if pp_intvw3 .in. {1}
                / select if age9 .in. {15- high};
block hicmpgrd4 / select if pp_intvw4 .in. {1}
                / select if age13 .in. {15- high};
block c_esr1 / class sex * race * ethnicity
      / select if c_pp_mis1 .in. {1}
      / select if c_age1 .in. {15-high};
block c_esr2 / class sex * race * ethnicity
      / select if c_pp_mis2 .in. {1}
      / select if c_age2 .in. {15-high};

... Similar block statements for months 3-12
```

The first BLOCK statements define the universe for highest grade completed but without an accompanying /CLASS specification. The subsequent BLOCK statements for the labor force status cross-classify by sex, race, and ethnicity. Although defined, the CLASS variable for TENURE is not associated with any of the BLOCK statements, and it is consequently excluded from the output VPLX file.

3.4 LABEL

By default, the label for each variable is its name, but more informative displays can be created by labeling key variables with labels of up to 24 characters. Exhibit 1.5 provided two example statements

```
label c_esr1 'Jan. 1987 Labor Force Status'
      c_esr2 'Feb. 1987 Labor Force Status'
```

```
          c_esr3 'Mar. 1987 Labor Force Status'
          c_esr4 'Apr. 1987 Labor Force Status'
          c_esr5 'May 1987 Labor Force Status'
          c_esr6 'Jun. 1987 Labor Force Status'
          c_esr7 'Jul. 1987 Labor Force Status'
          c_esr8 'Aug. 1987 Labor Force Status'
          c_esr9 'Sep. 1987 Labor Force Status'
          c_esr10 'Oct. 1987 Labor Force Status'
          c_esr11 'Nov. 1987 Labor Force Status'
          c_esr12 'Dec. 1987 Labor Force Status';

    label hicmpgrd1 'Completed ed., intvw 1'
          hicmpgrd2 'Completed ed., intvw 2'
          hicmpgrd3 'Completed ed., intvw 3'
          hicmpgrd4 'Completed ed., intvw 4' ;
```

Note that variable names and labels alternate, with a single ending semicolon. If an apostrophe is required, then two adjacent apostrophes are treated as a single apostrophe within the label (the same rule as for labels of levels of class and categorical variables stated in Section 3.2).

Label statements are not executable statements, that is, they do not cause any action to occur for each observation. The label is simply stored on the outgoing VPLX file. The variables named in the label statement must have been defined by previous statements, but their placement is otherwise arbitrary. They may be placed in the output section along with BLOCK statements, since they affect the information stored on the VPLX file and are thus concerned with output. In fact, they were placed for convenience near the corresponding CAT statement in the classification section in Exhibit 1.5.

3.5 BY

The BY feature of VPLX corresponds to a similar feature in SAS. Generally, it is advantageous to use this feature only for large problems. For small and moderate problems, the same purposes can usually be accomplished as effectively and with fewer complications through CLASS.

The BY statement classifies one or more variables as BY variables. The variables may either be real or class variables. In either case, the file must be in sort by the BY variables. In cases where the replication method also has an associated sort, such as the jackknife, the primary sort must be the BY variables, followed by the sort required by the replication method as the primary sort.

The syntax of the statement is simple,

```
by vlist ;
```

where *vlist* is a list of one or more variables to be declared as by variables. If more than one BY variable is used, then *vlist* should reflect the order of importance, as in,

```
   by state county tract block ;
```

The BY statement should appear in the categorization section of the CREATE step. The use of a variable as a BY variable is recorded on the VPLX file, and DISPLAY and other VPLX steps that might use the file will take this into account.

```
/* The input data are the 12 records of i3-3.dat:
 5 7 1 1 1 1
 6 8 1 2 1 1
 5 2 1 3 2 1
 4 1 1 4 2 1
 8 4 1 5 3 1
 8 2 1 6 3 1
 6 7 1 1 4 2
 7 8 1 2 4 2
 6 2 1 1 5 2
 5 1 1 2 5 2
 9 4 1 1 6 2
 9 2 1 2 6 2
*/

create  in = i3-3.dat  out = vplx1.vpl ;

  input  rooms persons  weight cluster
          stratum city /format (6f2.0) ;

  class rooms (4-5/6-high)             ! A class variable similar to
      '4-5 rooms' '6 or more rooms';   ! rooms1 in i3-2

  class city (1/2) 'New York'          ! city is made into a class variable
         'Los Angeles';

  by city;                             ! BY statement

  block persons / class rooms  ;       ! Note: this block statement is
                                       ! equivalent to the default option
                                       ! when no block statement is present

    Stratified jackknife replication assumed

    Size of block   1  =             4

    Total size of tally matrix =     4

**** End of CREATE specification/beginning of execution

    End of primary input file after obs #      12

         6 strata observed on incoming file

display                                  ! Beginning of DISPLAY step

list total (persons) /class rooms
```

```
    city                    :   New York

    rooms                   :   4-5 rooms
                                                    Estimate      Standard error

persons                 :   TOTAL                    10.0000              7.0711


    rooms                   :   6 or more rooms

                                                    Estimate      Standard error

persons                 :   TOTAL                    14.0000              8.2462

    city                    :   Los Angeles

    rooms                   :   4-5 rooms

                                                    Estimate      Standard error

persons                 :   TOTAL                     1.0000              1.0000

    rooms                   :   6 or more rooms

                                                    Estimate      Standard error

persons                 :   TOTAL                    23.0000              3.0000
```

Exhibit 3.4 An extract from i3-3.lis, showing a BY variable based on city. The CLASS statement for city associates a label for each value of city. Without the CLASS statement, the numerical results would have been the same but city would have been shown as 1.000 and 2.000 in the display. The DISPLAY step automatically produces results by city without additional instruction. The example is quite artificial, however, since more flexibility is available simply by leaving city is a class variable only.

In the example in Exhibit 3.4, VPLX examined the entire file, finding 6 strata, and then created the replicate samples reflecting all 6, even though only 3 were present in each city. In other situations, strata might appear in more than one BY group. In general, VPLX builds a consistent replication representation across the entire file. Volume II includes further technical detail on the use of BY.

The BY feature is used during part of the VPLX processing for the monthly Current Population Survey (CPS). One of the steps involves separate adjustments for each of 8 rotation groups in the survey. Since the calculations are of the same form but entirely separate for the rotation groups, it is advantageous to use the BY feature to restrict the size of the large arrays that must be considered to one rotation group at a time.

Another use of the BY feature is to carry out Monte Carlo studies, particularly of variance estimators. An independent program, written in Fortran or SAS, can generate multiple Monte Carlo samples. A single input file to VPLX may be built, containing perhaps thousands of replicate samples identified by a variable that may then be used as a BY variable. VPLX can compute estimates from each of the replicate samples and output them to a single file for further analysis, using VPLX, SAS, or other analytic tools.

NOTES

1. Section 1.6 enumerates the list of excluded variable names, all of which have some syntactic role. Inclusion of an excluded name on an input list or in other operations to define variables will generally be detected and treated as an error.

2. Although it would be convenient to couple the /SELECT IF syntax with the full range of logical expressions available for IF and ELSE IF, a syntactic complexity occurs in attempting to interpret "/" as division as part of an arithmetic expression or as a delimiter in the BLOCK statement. This problem does not occur with the .IN. { } syntax.

3. Although the SIPP example was selected to provide some flavor of larger applications, it still is rather modest. One measure of this is the use of double precision storage, reported by VPLX at the end of i1-3.lis as 133832 out of 1000000. Thus, there was reserve storage for problems over 7 times as large. In fact, VPLX can consider even much larger problems by making multiple passes of the data, although at the cost of longer calculation time. The program determines whether multiple passes are necessary; no specific instruction is required from the user.