2. How VPLX Works

2.1 Introduction

This chapter provides an overview of how VPLX uses replication to estimate variances for complex samples. The primary purpose of the chapter is to introduce each of the major replication methods offered by VPLX. Each method will be applied to a simple example problem. Chapter 12 discusses the available variants of each replication method in greater detail.

The discussion in this chapter will generally be limited to variance estimation for sampling with replacement. In practice, such methods are suitable even when the actual sampling method is without replacement, as long as the sample size is a small fraction of the population. (Chapter 12 considers sampling without replacement, including the instance when some of the elements of the sample are selected with certainty, that is, with probability 1.)

A secondary purpose of this chapter is to illustrate the general form of VPLX syntax and the interpretation of the resulting output. Subsequent chapters present the syntax systematically, so that the reader is not expected to deduce any of these rules from the examples appearing here. The command language and output are included simply to show the connection between each replication method and its implementation in VPLX and to provide an overall sense of the features of the program.

The chapter is not a substitute for formal training in variance estimation for complex samples. Ideally, each VPLX application should be implemented or reviewed by a statistician with previous experience in this area. Nonetheless, the chapter may serve as a place to start for some with limited backgrounds in inference from complex samples but who are aware that one or more applications of interest to them cannot be adequately approached with the statistical theory for simple random samples. A concluding section will discuss further references.

2.2 The Random Group Method

General Description. The random group method is perhaps the simplest replication method to understand, although its statistical properties make it probably the least attractive of those available in VPLX. In the random group method, the total sample is divided into parts, called random groups, in a manner designed to represent the major sources of variation arising from the sample design.

As a first example, consider a sample design drawing 30 clusters from a universe that has been divided into clusters of one or more associated elements each. For instance, a simple random sample of 30 households gives a clustered sample of persons, since each sampled household brings one or more associated persons into the sample. A possible application of the random group method in this instance is to treat the households as 30 random groups. Estimates of population characteristics are then formed based on each of the 30 sampled households separately, and then sampling variance is estimated based on variation among the 30 random group estimates. In general, when the sample is a simple random sample of clusters, one form of the random group method is to treat each of the sampled clusters as a random group.

Continuing the example, suppose that per capita income is one of the characteristics of interest. The estimate of per capita income from the full sample is the ratio of estimated aggregate income divided by the estimated number of persons. The random group method first applies the estimator for the full sample to the random group; in this case, the random group estimates are the household per capita incomes, that is, the total household income divided by the number of persons in the household. The random group variance estimate is then the familiar unbiased estimate of the variance of the mean of the resulting 30 values of household per capita income. In other words, once the random group estimates are determined for each random group, then the variance estimate may be easily obtained from standard statistical software, such as SAS, as well as more specific programs, such as VPLX. Note that the mean of the household per capita incomes may be quite far from the estimate of per capita income for the full sample.

The random group method assumes that it is possible to apply the full sample estimator to each individual random group. For example, if the population characteristic were mean family income, then single person households and other households without families pose a severe problem for the random group method when households are used as random groups.

Grouping Clusters. If the sample is composed of a simple random sample of a large number of sampled clusters, however, it is often preferable to group a number of clusters together in forming each random group. For example, a sample of 300 clusters may be divided into 100 random groups, each composed of 3 clusters. In grouping clusters together, it is important not to use observed characteristics in forming the random groups; instead, the grouping should be random or based simply on the order of selection. For instance, the first three sampled clusters could be grouped into a first random group, the next three into the second, etc. (On the other hand, it is almost never appropriate to divide sample clusters into elements and then to assign the elements to different random groups. Such assignment would misrepresent the clustered nature of the sample.)

Reducing the number of random groups by grouping has the disadvantage of producing generally less precise variance estimates for linear estimates, such as estimates of totals. Grouping has the

advantage of reducing the amount of calculation and, more importantly, may produce more reliable variance estimates for nonlinear statistics, such as ratios, since these nonlinear statistics are estimated from larger random group samples. Furthermore, judicious grouping may assure that the characteristics of interest are defined for each random group in situations in which some characteristics may be occasionally undefined at the level of the cluster. Nonlinear statistics based on single clusters may be highly unstable and present an unsatisfactory measure of variance for the estimate based on the full sample.

Only in relatively extreme circumstances should concerns about computer resources dictate a reduction in groups. For example, it will hardly ever be appropriate to reduce the number of random groups from 30 to 10 simply for fear of taxing VPLX, although such a grouping may be occasionally called for if complex nonlinear statistics would behave substantially better for the reduced number of replicates.

Stratification. In some situations, it is possible to reflect the effect of stratification in forming random groups. The simplest such situation is when the same number of clusters have been drawn independently from each of the strata. In this case, random groups may be formed by randomly assigning one of the sampled clusters from each stratum to a group. In other words, each random group would comprise a sampled cluster from each of the strata. For example, if there had been 3 strata with 10 sampled clusters each, a possible application of the random group method would divide the overall sample into 10 random groups, each with 1 sample cluster from each of the 3 strata.

As a general comment, however, the random group method does not adapt to stratification nearly so effortlessly as the stratified jackknife, to be discussed in Section 2.4. When stratification is present, it is generally important to reflect its effects in designing the random group samples. If the numbers of sampled clusters per stratum do not permit a satisfactory application of the random group method, then one should move toward more flexible alternatives, such as the stratified jackknife.

How Many Replicates? Applications of random group estimates have typically chosen a relatively modest number of random groups, such as 10 or 30. Generally, too small a number, such as 2, produces highly unstable variance estimates, while large numbers, such as 500, run a high risk of difficulties for nonlinear estimators.

Example. A simple numerical example will precede the formulas for the random group method. A file of commands, `exam1.crd` , for the example contains:

```
comment   EXAM1
comment   First example input into VPLX
```

```
          This example starts with two variables, rooms and persons.

          There are six observations, each treated as a separate
          sampling cluster.  This run illustrates the random group
          method.

          The create step organizes the incoming data into replicate
          totals for use by later steps.
create  in = example1.dat  out = example1.vpl
input    rooms persons cluster
format   (3f2.0)
comment  The incoming data, in the file example1.dat, are:
 5 7 1
 6 8 2
 5 2 3
 4 1 4
 8 4 5
 8 2 6
replication method random group
replicate number    cluster
display  in = example1.vpl
comment  The display step takes the VPLX file created in the
         previous step and displays estimates, standard errors,
         etc. as requested.
list     rooms  persons  total(rooms persons)
cov      total(rooms  persons)
```

Exhibit 2.1. First example input to VPLX, illustrating the random group method.

This example problem was run in DOS with the command:

```
c>vplx  <exam1.crd >exam1.lis
```

where `exam1.crd` was the name of the file in Exhibit 2.1 and `exam1.lis` became the output file from VPLX.

Some useful generalizations may be made about the VPLX syntax from Exhibit 2.1. VPLX commands begin with a key word in position 1 and may be continued on an arbitrary number of records as long as the first character of each continuation record is a blank; in other respects the command language is free format. (Chapter 7 discusses optional indentation of key words by placing "_" in the first position.)

The example includes two steps, as described in Chapter 1, a CREATE step and a DISPLAY step. Comments may be interspersed among the rest of the commands.

The resulting output, `example1.lis` , contained:

VPLX - Version 92.12

```
comment   EXAM1

comment   First example input into VPLX.

          This example starts with two variables, room and persons.

          There are six observations, each treated as a separate
         sampling cluster.  This run illustrates the random group
          method.


          The create step organizes the incoming data into replicate
          totals for use by later steps.

create  in = example1.dat  out = example1.vpl                              #1

input    rooms persons cluster

      3 variables are specified                                            #2

format   (3f2.0)

comment  The incoming data, in the file example1.dat, are:
 5 7 1
 6 8 2
 5 2 3
 4 1 4
 8 4 5
 8 2 6

replication method random group                                            #3

replicate number    cluster

    Size of block   1  =              3                                    #4

    Total size of tally matrix =      3

    Unnamed scratch file opened on unit 13

    Unnamed scratch file opened on unit 14

    End of primary input file after obs #       6


display  in = example1.vpl                                                 #5


comment  The display step takes the VPLX file created in the
         previous step and displays estimates, standard errors,
         etc. as requested.


list     rooms  persons  total(rooms persons)                              #6

cov      total(rooms  persons)
```

```
                                                Estimate      Standard error        #7

rooms                    :  MEAN                   6.0000              .6831

persons                  :  MEAN                   4.0000             1.1832

rooms                    :  TOTAL                 36.0000             4.0988

persons                  :  TOTAL                 24.0000             7.0993

                      Covariances of the Sample Estimates                          #8

                                    Estimate              1              2


    rooms                :  TOTAL
   1                                 36.0000   .16800000D+02

    persons              :  TOTAL
   2                                 24.0000   .12000000D+01   .50400000D+02

Use of double precision matrix:      35 out of   16000                            #9
Stop - Program terminated.
```

Exhibit 2.2. Annotated output from the first random group example.

The output from VPLX was modified for inclusion in Exhibit 2.2 by removing page breaks inserted by VPLX and adding numbered comments, #1, #2, etc. to the right margin.

The output begins with a line showing a version number, which indicates the year and month of the last major changes to the program. VPLX echoes back all of the input commands. (Consequently, subsequent examples in this chapter will show only the output without the associated input.) VPLX adds comments, such as those at #2 and #4, and also displays results, such as those beginning at #7 and #8. (For purposes of this documentation, the material reported by VPLX at #2 and #4 is shown in italics to identify it as separate from the input commands. Obvious output from the program, beginning at #8 and #9, is not so identified, however.)

The first set of comments in Exhibit 2.2 note that the example is based on a hypothetical situation in which a simple random sample of 6 occupied housing units has been sampled from some domain of interest, such as a city. (As noted at the beginning, the chapter considers only variance estimators for sampling with replacement, although VPLX is also able to reflect finite population corrections in some applications; see Chapter 12.) For simplicity, unweighted estimates are considered, although VPLX can readily incorporate weights. Two variables are obtained for the

sample cases: the number of rooms in the housing unit and the number of persons in the household.

The first statement of the CREATE step appears at #1. The statement specifies two files: a data file, `example1.dat` , described in the following comment; and an output file, `example1.vpl` , that VPLX will create in this step. The output file contains variable identifiers, characteristics, and estimated totals ("tallies") necessary to carry out replicate calculations. The CREATE statement is followed by two required statements: an INPUT statement naming three variables to be read, and a FORMAT statement giving the locations of the variables on the incoming file. The FORTRAN format (3f2.0) states that each of three variables occupy two positions each, beginning at the start of the record, and that there are no implied places after the decimal. VPLX reports the count of variables at #2 to assist in checking the consistency of the variable list and format. Using the format, VPLX reads the first incoming record as 5., 7., and 1., and reads subsequent records in a similar manner. INPUT and FORMAT statements must appear in any CREATE step.

The statement at #3 serves to identify the replication method as random group, and the following statement identifies one of the incoming variables, `cluster` , as the random group identifier. In this simple example, each separate observation is treated as a random group, although the more usual situation is that a random group is composed of several sample observations.

VPLX produces the report at #4, comprising five indented lines, in the course of carrying out the CREATE step. The first two lines report on the number of cells used to hold the sums for each replicate. In this case, three cells are required for each replicate to hold the two variables, `rooms` and `persons`, and a third for the number of cases. (In a weighted analysis, the last of these would be the weighted number of cases.) Since the variable `cluster` is used to identify random groups, no sum is made for it. There is only one block created, so that the total size for each replicate is 3. The next two lines report that two scratch files have been opened. The last reports that 6 observations have been successfully read from the primary input file.

The DISPLAY step begins at #5. The step reads the VPLX file, `example1.vpl` , created in the previous step. The LIST statement at #6 requests estimates and standard errors for the means of `rooms` and `persons` , as well as their respective estimated totals. (Because the sample size was fixed at 6, the unweighted totals are less meaningful than when weighted estimates are considered, since the weighted values estimate population totals. Nonetheless, it is useful to consider the unweighted totals here for purposes of illustration.) The COV statement immediately afterwards requests the covariance matrix for total rooms and persons.

The output at #7, normally beginning on the top of a page, presents the estimates requested by the LIST statement. For example, the estimated mean number of rooms is 6, with an estimated

standard error of .6831. Note that this value represents a measure of the uncertainty of the estimate 6.0000 as an estimate of the population mean number of rooms, rather than an estimate of the population standard deviation of the number of rooms. Except for rounding, the standard error of the total number of rooms is 6 (that is, the number of cases or weighted cases) times the standard error for the mean. In general, however, there is not a simple relationship between the standard errors of the mean and total when there is variation in the number of sample cases per random group.

The output at #8 displays the estimated covariance matrix for the two estimated totals. One column again shows the estimates. The covariance of total rooms with itself, that is, its variance, is .168D+2 = 16.8, the square of the standard error, 4.0998, shown previously for the LIST request. Similarly, the other diagonal element, .504D+02 = 50.4, can also be computed from the output corresponding to the LIST statement. The covariance between total rooms and persons, .12D+01 = 1.2, cannot be derived from the values obtained from the previous LIST statement, however.

When VPLX concludes processing, it gives a summary, at #9, of its use of a double precision storage matrix. Most data storage for a problem is obtained from this array, and VPLX will terminate and provide an error message if insufficient storage is available. The size of this matrix can vary with the computer environment: the 93.05 PC version, compiled with the Microsoft Fortran 5.0 compiler, has 16,000 cells available, but in a VAX environment, for example, the array can be usefully set to 1,000,000 or more. (The Microsoft FORTRAN Workbench compiler permits use of extended memory on the PC, and version 93.05 has been compiled and run with 160,000 cells.) In this example and all the others in this chapter, demands on storage are extremely small compared to the storage available on the PC version. For the sake of brevity, this concluding summary will be omitted from the subsequent examples shown in this chapter, as will the version number displayed by VPLX at the beginning of the output.

This first example illustrates that with some simple commands it is readily possible to obtain sampling errors and covariances for means and totals through a combination of CREATE and DISPLAY steps. Through simple elaboration of the INPUT and FORMAT statements and the requests in the DISPLAY step, it is possible to obtain results for a much larger set of variables in a single run.

The TRANSFORM Step. To enrich the example to show other capabilities of VPLX, however, suppose that the ratio of rooms to persons, computed on an aggregate basis, were also of interest. Ratios of sample values routinely appear in survey analysis. The TRANSFORM step can read a VPLX file and compute ratios and other such statistics defined in terms of estimated sample totals for both the overall sample and each replicate sample, and writes the results to a new VPLX file. The DISPLAY step is then able to provide estimates and variances for such estimates. A

set of commands, `exam2.crd` , to VPLX to accomplish this goal for the same data set produced the following output:

```
comment   EXAM2


comment   This next example again uses the random group method but also
          uses it to compute the variance of the ratio of the number
          of rooms to the number of persons.  The initial create
          step is the same as EXAM1, and uses the same input.

create  in = example1.dat  out = example1.vpl

input    rooms persons cluster

      3 variables are specified

replication method random group

replicate number    cluster

format     (3f2.0)

    Size of block   1  =              3

    Total size of tally matrix =      3

    Unnamed scratch file opened on unit 13

    Unnamed scratch file opened on unit 14

    End of primary input file after obs #       6

transform  in = example1.vpl out=exampl1a.vpl                                  #1


comment   The transform step is able to take a VPLX file and to create
          or modify statistics that are functions of sample totals
          such as the ratio of total rooms to total persons.  The next
          statement, divide, is followed by statements instructing VPLX
          to divide rooms by persons and to place the result in proom.

divide                                                                         #2

old  rooms persons

derived   proom

    (assigned to block   2)                                                    #3


comment   The next call is to a subroutine, rprint, that prints the
          results.  Note that it is called once for the full sample and
          once for each replicate.

rprint
```

2.10

```
old rooms persons proom

comment   Optionally, labels can be provided and added to the VPLX file.

labels  rooms 'Number of rooms' persons 'Persons'
        proom   'Rooms per person'

RPRINT: REPLICATE    0                                                          #4
            rooms       :  TOTAL          36.000
            persons     :  TOTAL          24.000
            proom       :  VALUE          1.5000

RPRINT: REPLICATE    1
            rooms       :  TOTAL          30.000
            persons     :  TOTAL          42.000
            proom       :  VALUE           .7143

RPRINT: REPLICATE    2
            rooms       :  TOTAL          36.000
            persons     :  TOTAL          48.000
            proom       :  VALUE           .7500

RPRINT: REPLICATE    3
            rooms       :  TOTAL          30.000
            persons     :  TOTAL          12.000
            proom       :  VALUE          2.5000

RPRINT: REPLICATE    4
            rooms       :  TOTAL          24.000
            persons     :  TOTAL          6.0000
            proom       :  VALUE          4.0000

RPRINT: REPLICATE    5
            rooms       :  TOTAL          48.000
            persons     :  TOTAL          24.000
            proom       :  VALUE          2.0000

RPRINT: REPLICATE    6
            rooms       :  TOTAL          48.000
            persons     :  TOTAL          12.000
            proom       :  VALUE          4.0000



display  in = exampl1a.vpl                                                      #5


comment  This will now display the random group estimate of the variance
         of proom.  Note that linearization gives:
         var(proom) = (1/24*24) * var(total(room))
                      + (36*36/24*24*24*24) * var(total(persons))
                      - 2 * (36/24*24*24) * cov (total(room),total(persons))
                    =  .2198  = .4688 * .4688

list     rooms  persons proom
```

			Estimate	Standard error
Number of rooms	:	MEAN	6.0000	.6831
Persons	:	MEAN	4.0000	1.1832
Rooms per person	:	VALUE	1.5000	.7055

Exhibit 2.3. Annotated output from the second random group example.

The VPLX run in this second example begins with a CREATE step with the same commands as the previous example, recreating the VPLX file `example1.vpl` . The second example could have begun instead at #1 as long as `example1.vpl` was still available. The TRANSFORM step at #1 reads the VPLX file `example1.vpl` as input and creates another VPLX file, `exampl1a.vpl`, as an output. Generally, the TRANSFORM step operates on an incoming VPLX file to produce a new one.

The TRANSFORM step creates new variables by using subroutines; in turn, the subroutines are of two types: standard and user-supplied. There are several standard subroutines available in the TRANSFORM step. EXAM2.CRD evokes two of these, DIVIDE, at #2, and RPRINT, a few lines below. Chapters 8 and 14 describe many of the available standard subroutines. The intention is that the standard routines should provide a means to accomplish many of the tasks that one might require, but the ability to include user-supplied routines affords a means of estimating variances for unusual problems.

The two lines following the DIVIDE statement at #2 determine what variables will be passed to and received from the DIVIDE subroutine. The first statement identifies `rooms` and `persons` as the two existing variables to be passed to DIVIDE. DIVIDE uses the last old variable in the list as the denominator; hence, the order of variables in the statement is important. The next statement identifies a new variable to be created by DIVIDE and declares its type as "derived." This variable type is used for statistics, such as ratios, that are not simply the weighted sum of individual observed values.

VPLX wrote the line at #3 in Exhibit 2.3 to state the block assignment for the new variable. The issue of blocking is discussed in Chapter 5. Let it suffice to note that VPLX handles the blocking of variables in the examples in this chapter without requiring from the user any specific direction in the command language.

A comment then notes that labels can be assigned to the variables and this operation then follows. This feature is also available within the CREATE step.

The lines beginning at #4 in Exhibit 2.3 were written by RPRINT, and the information in these lines allows us to look directly at the manner in which random group replication is implemented in VPLX. The first set of lines shows values for `rooms`, `persons`, and their ratio `proom` as 36, 24, and 1.5, respectively, which are the values for the entire sample. In other words, VPLX first calls the subroutine with the values for the overall sample. The next group of lines reports values for rooms and persons of 30 and 42, equal to 6 times the values for the first random group, 5 and 7 respectively. (The values on the incoming file are reported after #1 in Exhibit 2.2.) Similarly, sample totals, computed as 6 times the random group values, are displayed in succession for the remaining random groups.

At #5, the DISPLAY step begins, using the new file, `exampl1a.vpl`, as input. Standard errors are obtained for the means of `rooms` and `persons`, as well as for the value of their ratio. If X_0 denotes an estimate computed for the whole sample, and X_r represent the same statistic based on random group r, r=1,..,6, then the random group variance estimator implemented by VPLX in this case is:

$$Var_{rg}(X_0) = \frac{1}{n(n-1)} \sum_{r=1}^{n} (X_r - X_0)^2, \qquad (2.1)$$

where *n* is the number of random groups. Note that, with the combination of circumstances in this example, this variance estimator gives identical results for the means and totals of rooms and persons as the classical text-book variance estimator for sampling with replacement.

Since the ratio `proom` is a nonlinear statistic, however, different variance estimation strategies yield differing results. For purposes of comparison, the comment after #5 gives the variance estimate for `proom` based on linearization, as implemented in several other programs, equivalent to an estimated standard error of 0.4688. With the random group estimator, VPLX obtains the estimated variance of `proom` as $1/30\ \{(0.7143-1.5)^2 + (0.75-1.5)^2 + (2.5-1.5)^2 + (4.0-1.5)^2 + (2.0-1.5)^2 + (4.0-1.5)^2\} = 0.7055^2$. Both linearization and random group methods can be applied to other smooth, although potentially complex, functions of the estimated sample totals.

The values of `proom` computed for the random groups vary widely, from .7143 to 4.0. In general, the random group approach recomputes estimates of interest based on a relatively small fractions of the original sample, and nonlinear statistics such as ratios may be far less stable for the random groups than for the whole sample. The next section considers another method, which yields replicate samples that resemble the overall sample much more closely.

Other requests made in the DISPLAY step in Exhibit 2.2 were not repeated in Exhibit 2.3, but could have been. In other words, it is possible to use the TRANSFORM step to increase the information on the VPLX file without losing any results from the CREATE step. Alternatively,

it is also possible to direct VPLX to retain selectively information from the CREATE step on the output file.

User-Supplied Subroutines in the TRANSFORM Step. Exhibit 2.3 employs only standard subroutines in the TRANSFORM step, which is illustrative of almost all practice. In some cases, however, the standard subroutines are not adequate for an application. User-supplied FORTRAN subroutines may be linked directly with VPLX in such circumstances. An alternative approach, in `exam3.crd`, employs a user-supplied subroutine, USER2, as an example. In general, up to 10 user-supplied FORTRAN subroutines, with names USER1 - USER10, may be linked with VPLX and called from the TRANSFORM step. Furthermore, because USER2 provides a more concise display from the TRANSFORM step than RPRINT, the balance of the chapter will employ it. The output of the TRANSFORM step in this case is:

```
transform  in = example1.vpl out=exampl1a.vpl

comment    The next statement, user2, requests that VPLX call a FORTRAN
           subroutine USER2, which can be provided by the user to
           manipulate the totals.
           Note that VPLX will call the routine once for the total and once
           for each replicate, and that USER2, for didactic purposes,
           displays the three values

user2

old  rooms persons

derived   proom

    (assigned to block   2)


comment   Optionally, labels can be provided and added to the VPLX file.

labels  rooms 'Number of rooms' persons 'Persons'
        proom   'Rooms per person'
REPLICATE  0, V1=   36.00, V2=   24.00 RATIO=  1.5000
REPLICATE  1, V1=   30.00, V2=   42.00 RATIO=   .7143
REPLICATE  2, V1=   36.00, V2=   48.00 RATIO=   .7500
REPLICATE  3, V1=   30.00, V2=   12.00 RATIO=  2.5000
REPLICATE  4, V1=   24.00, V2=    6.00 RATIO=  4.0000
REPLICATE  5, V1=   48.00, V2=   24.00 RATIO=  2.0000
REPLICATE  6, V1=   48.00, V2=   12.00 RATIO=  4.0000
```

Exhibit 2.4. Alternative TRANSFORM step based on USER2.

The comment after the TRANSFORM statement explains that VPLX will call a FORTRAN subroutine named USER2 to obtain values of the ratio. The FORTRAN code for USER2 in this example is:

```
      SUBROUTINE USER2(DID,NID,IVLIST,DIMPNT,DIMX,DXPNT,                    #1
     .  DX,MSIZE,MTYPE,MTRAN,STRING,IX)
      INTEGER NID,IVLIST(6),DIMPNT(*),DIMX(*),DXPNT(*),MSIZE(*),
     . MTYPE(*),MTRAN(*),IX(*)
      DOUBLE PRECISION DID(NID),DX(*)
      CHARACTER*128 STRING(*)
  100 FORMAT(' REPLICATE',I3,', V1=',F8.2,', V2=',F8.2,' RATIO=',F8.4)  #2
      IF(DX(DXPNT(2)).GT.0.) THEN
        DX(DXPNT(3))=DX(DXPNT(1))/DX(DXPNT(2))                          #3
      ELSE
        DX(DXPNT(3))=0.
      END IF
      K=DID(1)+.05D0
      WRITE(6,100)K,DX(DXPNT(1)),DX(DXPNT(2)),DX(DXPNT(3))              #4
      RETURN
      END
```

Exhibit 2.5. Example FORTRAN subroutine USER2 in the TRANSFORM step.

The argument list to USER2, at #1 in Exhibit 2.4, provides information to the subroutine, most of which is not referenced in this example. A few observations, even to the reader unfamiliar with FORTRAN syntax, are instructive. Chapter 15 describes how such subroutines may be written and used. The `FORMAT` statement at #2 is used, in connection with the `WRITE` statement at #4, to write the replicate number and values of the variables directly to the listing. More typically, the user-supplied routines will have only computational objectives, but, writing values to the listing file and all of the other operations available in FORTRAN are permitted, with the exception of altering the files that VPLX has open or the contents of VPLX's common blocks.

The statement at #3 in Exhibit 2.4 computes the ratio. Note that the array `DXPNT` points to locations in `DX` for values of the variables. In general, it is possible to create new statistics based on arbitrarily complex functions of the estimated totals as long as they can be implemented in user-supplied subroutines.

2.3 The Simple Jackknife

General Description. The simple jackknife can be applied in essentially the same sampling situations as the random group method. Whereas the random group method uses a small fraction of the sample to compute all the statistics of interest, the jackknife leaves one of the fractions out of the estimate of total, in succession. In other words, instead of replicate estimates based on only one group, the jackknife creates replicate estimates based on all but one group.

In general, a data file suitable for VPLX to implement the random group method is equally suitable for the jackknife - VPLX itself forms the jackknife replicates leaving out one group at a time.

Grouping Clusters. The rules for the random group method about grouping clusters, in Section 2.2, may be applied to the jackknife as well. The incentives for grouping are substantially less with the jackknife, however. Since the jackknife uses almost all of the sample to compute the statistics of interest, the concerns for the stability of nonlinear statistics for the random group method greatly diminish for the jackknife.

Stratification. As with the random group, the simple jackknife may sometimes be adapted to problems involving stratification. In most instances, however, it is more advantageous to switch to the stratified jackknife, in Section 2.5, for any such problems.

How Many Replicates? Many users may be surprised at how fast VPLX implements a jackknife calculation - almost as quickly as a random group method. In most instances, it is simply more convenient for VPLX to work with up to hundreds of replicate samples on a PC or thousands on workstations or higher level computers before considering grouping clusters for increased efficiency. Thus, in many instances, the appropriate number of replicates is the number of clusters in the design.

On occasion, however, there are reasons to reduce the number of replicates by appropriate grouping. The VPLX file generated by the CREATE step or subsequent TRANSFORM step may be too large for available storage space if the jackknife employs all available clusters. Secondly, a very large number of replicates - in the thousands on a PC, for example - may represent too great a demand on resources.

There is not an absolute rule on the minimum number of replicates, although variance estimates based on 20 replicates are generally only moderately precise, and using a larger number, such as 100, if possible, is almost always worth the effort in increased precision for the variance estimates.

Example. Reanalyzing the previous example with the jackknife is instructive:

```
comment  EXAM4


comment  The next calculation uses the same data but substitutes
         the simple jackknife method.  Note that by specifying
         a cluster code among the input variables and no stratum
         code, the simple jackknife is selected by default.

create  in = example1.dat  out = example1.vpl                              #1
```

```
input    rooms persons cluster

      3 variables are specified

format   (3f2.0)

    (Simple) jackknife replication assumed

    Size of block   1  =              3

    Total size of tally matrix =      3

    Unnamed scratch file opened on unit 13

    Unnamed scratch file opened on unit 14

    End of primary input file after obs #       6


transform  in = example1.vpl out=exampl1a.vpl

user2

comment  Notice that the simple jackknife forms replicates by
         leaving out one cluster at a time.  The values worked
         with by USER2 are quite different from the random group
         method.

old  rooms persons

derived   proom

    (assigned to block   2)

labels  rooms 'Number of rooms' persons 'Persons'
        proom  'Rooms per person'
REPLICATE  0, V1=   36.00, V2=   24.00 RATIO=  1.5000                          #2
REPLICATE  1, V1=   37.20, V2=   20.40 RATIO=  1.8235
REPLICATE  2, V1=   36.00, V2=   19.20 RATIO=  1.8750
REPLICATE  3, V1=   37.20, V2=   26.40 RATIO=  1.4091
REPLICATE  4, V1=   38.40, V2=   27.60 RATIO=  1.3913
REPLICATE  5, V1=   33.60, V2=   24.00 RATIO=  1.4000
REPLICATE  6, V1=   33.60, V2=   26.40 RATIO=  1.2727



display  in = exampl1a.vpl

list     rooms  persons total(rooms persons) proom


comment  Note that the covariances for the totals are the same
         as the random group method.

cov      total(rooms  persons)
```

```
                                                    Estimate      Standard error

Number of rooms         :  MEAN                       6.0000              .6831

Persons                 :  MEAN                       4.0000             1.1832

Number of rooms         :  TOTAL                     36.0000             4.0988

Persons                 :  TOTAL                     24.0000             7.0993

Rooms per person        :  VALUE                      1.5000              .5220

                      Covariances of the Sample Estimates

                                    Estimate              1              2

 Number of rooms         :  TOTAL
1                                    36.0000   .16800000D+02

 Persons                 :  TOTAL
2                                    24.0000   .12000000D+01   .50400000D+02
```

Exhibit 2.6. Annotated output from the simple jackknife example.

At #1, a comparison to Exhibit 2.2 shows that the syntax for the simple jackknife in the CREATE step is similar to the random group method, but even simpler. Use of `cluster` as a variable name carries the reserved meaning that the variable is to be used to form clusters for the jackknife. With this much information, and without an explicit identification of the replication method through a REPLICATION METHOD statement, VPLX will carry out a simple jackknife.

The statements for the TRANSFORM step are the same as for the random group example, but the output of USER2, appearing at #2, is quite different. The first set of values corresponds to the values for the overall sample, as before. The first replicate is computed by omitting the value for the first cluster and reweighting the remaining results by 6/5, that is, 37.2 = 6/5 (36-5). The current range of values for the ratio, 1.2727 to 1.8750, is much narrower than for the random group method.

VPLX implements the following formula for the simple jackknife:

$$Var_{jk}(X_0) = \left(\frac{n-1}{n}\right) \sum_{r=1}^{n} (X_r - X_0)^2, \qquad (2.2)$$

where n is the number of clusters. This formula is similar to (2.1), but the coefficient $(n\text{-}1)/n$ in (2.2) is $(n\text{-}1)^2$ times larger than the analogous coefficient in (2.1), reflecting the much narrower

range of variation in replicate estimates for the jackknife than the random group. Variances and covariances for means and totals of `rooms` and `persons` are identical with random group results in this example. In general, the jackknife and random group procedures agree for linear statistics when computed from the same set of clusters.

Generally, the random group and jackknife disagree for nonlinear statistics. The jackknife estimates the variance for `proom` as $5/6\ \{(1.8235-1.5)^2 + (1.8750-1.5)^2 + (1.4091-1.5)^2 + (1.3913-1.5)^2 + (1.4-1.5)^2 + (1.2727-1.5)^2\} = 0.5220^2$. The jackknife variance estimate is relatively close instead to the linearized version of 0.4688^2. In general, the jackknife tends to be close to the linearized variance estimate if both calculations employ the same clusters and the statistic of interest is smooth.

Applications of the jackknife are safest for smooth statistics, for example, statistics with continuous first and second derivatives in a neighborhood of the population value. (It is not necessary that the user be able to compute these derivatives, simply that the derivatives exist theoretically.) Means, totals, proportions, and a variety of ratio, regression, and analytic statistics fit this description. Sample medians and other percentiles (except when computed through extrapolation of grouped data) do not. Statistics whose form changes abruptly based on characteristics of the sample data, such as ratio adjustments that collapse based on rules applied to ratios observed in the sample data, or model selection rules in linear regression, also pose problems for the jackknife. (Typically, these situations are problematic for linearization as well, although there are specific approaches available for percentiles and some other statistics.) To some degree, these problems can be overcome by grouping, but the user should consult the developing research in applications of the jackknife to such statistics with jumps or other discontinuities.

2.4 The Stratified Jackknife

General Description. Many complex samples employ stratification in the sample design, that is, they divide the universe into distinct subpopulations and sample each subpopulation separately and generally independently. Both the random group method and the simple jackknife adapt with difficulty to stratification, except under special circumstances. When two or more clusters have been sampled from each of two or more strata, VPLX is able to implement a stratified jackknife, which is more suitable for such problems.

The simple jackknife of the previous section omitted a cluster and reweighted the remaining n-1 clusters. In the stratified jackknife, a cluster in stratum s is omitted, and the remaining n_s-1 clusters within stratum s are reweighted. Thus, the stratified jackknife assumes that a given

cluster represents the stratum from which it was selected, not the population as a whole. As long as $n_s>1$ for each stratum s, the stratified jackknife does not impose further restrictions on the n_s.

Grouping Clusters. If necessary, clusters may be grouped in two ways: 1) clusters within a stratum may be grouped following the suggestions given in Sections 2.2 and 2.3 for the random group and simple jackknife methods; 2) in some situations, it is possible to group clusters across strata. As an example of the second situation, suppose a design includes *k* strata with 2 sampled clusters each. Suppose further that these strata contribute an extremely small proportion of the overall estimate and variance. Then by randomly selecting one cluster from each of the strata to form one super-cluster and assigning the remaining clusters to the second, one may include the resulting two super-clusters and super-stratum in the analysis in place of the original.

How Many Replicates? VPLX implements the stratified jackknife almost as quickly as the simple jackknife. Consequently, relatively large numbers of clusters and strata may be considered.

Pronouncements about how few replicates are adequate become more complex for the stratified jackknife for two reasons. First, an algebraic degree of freedom is lost for each stratum, so that a stratified jackknife based on 50 clusters, 2 clusters from each of 25 strata, behaves more like a simple jackknife using 26 clusters rather than a simple jackknife using 50, with respect to the relative reliability of the variance estimate. Secondly, the stratified jackknife becomes even less precise if most of the true variance for a given statistic is due to sampling from only one of the strata, if only a few clusters from the stratum are present for purposes of variance estimation. Consequently, it is advisable to consider providing VPLX with all of the available clusters in the design before grouping clusters, if at all possible.

Example. Suppose that the data from the previous example had not been drawn by sampling with replacement from the overall population but instead was the result of sampling from a population after stratification into 3 strata, from each of which two occupied households had been selected. For example, the first two sample units may have been sampled from a stratum of housing units in multi-unit buildings in areas with high poverty rates, the second two from a stratum of housing units in multi-unit buildings in other areas, and the last two from a stratum of single-unit dwellings. The random group method or the jackknife could be applied to this problem by creating two replicates based on assigning one of each pair of sample units within each stratum to the first replicate and the remaining three to the second. (In this instance, where there are only two replicate samples, the random group and jackknife replication are equivalent.) A better approach is based on the stratified jackknife:

```
comment   EXAM5

comment   The next example illustrates the stratified jackknife
```

```
          The six observations are grouped into three strata of
          two observations each.

create  in = example5.dat  out = example5.vpl

input    rooms persons cluster stratum / option nprint = 6                    #1

      4 variables are specified

format   (4f2.0)

comment   EXAMPLE5.DAT contains the following data:                           #2
 5 7 1 1
 6 8 2 1
 5 2 3 2
 4 1 4 2
 8 4 5 3
 8 2 6 3

    Stratified jackknife replication assumed                                  #3

    Size of block   1  =             3

    Total size of tally matrix =      3

    Unnamed scratch file opened on unit 13                                    #4

    Unnamed scratch file opened on unit 14

    Unnamed scratch file opened on unit 15

Observation     1 from unit 12
     rooms              5.0000         persons            7.0000
     cluster            1.0000         stratum            1.0000

Observation     2 from unit 12
     rooms              6.0000         persons            8.0000
     cluster            2.0000         stratum            1.0000

Observation     3 from unit 12
     rooms              5.0000         persons            2.0000
     cluster            3.0000         stratum            2.0000

Observation     4 from unit 12
     rooms              4.0000         persons            1.0000
     cluster            4.0000         stratum            2.0000

Observation     5 from unit 12
     rooms              8.0000         persons            4.0000
     cluster            5.0000         stratum            3.0000

Observation     6 from unit 12
     rooms              8.0000         persons            2.0000
     cluster            6.0000         stratum            3.0000
(Printing discontinued on unit 12)

    End of primary input file after obs #       6
```

```
         3 strata observed on incoming file

transform  in = example5.vpl out=exampl5a.vpl

user2

old  rooms persons

derived   proom

    (assigned to block   2)

labels  rooms 'Number of rooms' persons 'Persons'
        proom  'Rooms per person'
REPLICATE  0, V1=   36.00, V2=   24.00 RATIO=  1.5000                          #5
REPLICATE  1, V1=   37.00, V2=   25.00 RATIO=  1.4800
REPLICATE  2, V1=   35.00, V2=   23.00 RATIO=  1.5217
REPLICATE  3, V1=   35.00, V2=   23.00 RATIO=  1.5217
REPLICATE  4, V1=   37.00, V2=   25.00 RATIO=  1.4800
REPLICATE  5, V1=   36.00, V2=   22.00 RATIO=  1.6364
REPLICATE  6, V1=   36.00, V2=   26.00 RATIO=  1.3846


display  in = exampl5a.vpl

list     rooms  persons total(rooms persons) proom


comment  Note that linearization gives the variance of proom at                #6
         .1284 * .1284

cov      total(rooms  persons)

                                                Estimate      Standard error

Number of rooms         :  MEAN                   6.0000               .2357

Persons                 :  MEAN                   4.0000               .4082

Number of rooms         :  TOTAL                 36.0000             1.4142

Persons                 :  TOTAL                 24.0000             2.4495

Rooms per person        :  VALUE                  1.5000               .1297


                      Covariances of the Sample Estimates

                                      Estimate              1              2


    Number of rooms         :  TOTAL
   1                                   36.0000   .20000000D+01

    Persons                 :  TOTAL
   2                                   24.0000   .20000000D+01   .60000000D+01
```

Exhibit 2.7. Annotated output from the stratified jackknife example.

The INPUT statement at #1 now lists four variables. The last of these is `stratum`, which VPLX will interpret as a stratum code in the absence of other information. This example also illustrates a general feature available in Version 93.05: an option to specify a number of observations to list in order to check whether the values are being correctly read from the incoming file.

The FORMAT statement reflects a modification to represent the four variables. The comment at #2 shows the input data set. Sample records in the same stratum receive the same stratum code. The cluster code has not been revised from EXAMPLE1.DAT. Although the example seems to suggest that it is necessary to number cluster codes continuously from 1, VPLX only uses the cluster code within the stratum as an identifier. Consequently, the cluster codes could instead have been 1, 2, 1, 2, 1, 2, or many other possible combinations, with the same effect.

The information reported by VPLX at #3 also differs from the previous example. At #4, it is evident that VPLX uses one more scratch file for this method than the simple jackknife. The purpose of the additional file is to hold temporary totals for the separate strata.

The optional display of incoming data occurs next. Usually, such a listing for even a few of the first records on the file should help to identify any mismatch between the INPUT list and the FORMAT.

The values of each replicate reported at #5 illustrate the manner in which the stratified jackknife forms replicates. The value for the replicate number of rooms, 37.0, arises by omitting the first value, 5, from the total, 36, and reweighting the remaining observation within the stratum, 6, by the factor 2. The stratified jackknife reweights observations within the same stratum to compensate for omitting a cluster, rather than reweighting all of the remaining observations as the simple jackknife does. If an observation came from a stratum of three clusters, the stratified jackknife would reweight the remaining two by 1.5, etc.

The note at #6 reports the result of a calculation similar to that just after #5 in Exhibit 2.3, but using the estimated variances and covariance for the stratified sample, given as the last part of Exhibit 2.7. That is, $.1284^2 = 2./24^2 + 6.*36^2/24^4 - 2*2.*36/24^3$. The estimated standard error from the stratified jackknife, .1297, is quite close to the linearization result. Note that both results, which consider the sample to be stratified, are dramatically different from the previous examples.

The variance estimator in VPLX for the stratified jackknife is:

$$Var_{sjk}(X_0) = \sum_{s=1}^{S} \left(\frac{n_s - 1}{n_s} \right) \sum_{i=1}^{n_s} (X_{is} - X_0)^2, \qquad (2.3)$$

where *s* indexes the strata, *S* is the total number of strata, *i* represents an index of the cluster within the stratum, and n_s represents the number of clusters in stratum *s*. Double indexing is shown here for clarity, but VPLX in fact internally considers the replicates to be subscripted by a single index.

2.5 Half-Sample Replication

General Description. Half-sample replication, which under some conditions is also called balanced repeated replication, uses replicates formed from half of the sample. For example, if a sample design consists of a number of strata from which two clusters have been selected (technically, selected independently and with replacement), then a half-sample replicate can be formed by selecting one sample cluster from each of the strata. Variance estimation through half-sample replication is more efficient if the assignment to half samples is done by the user in an orthogonal manner, such as may be achieved through application of a Hadamard matrix; further technical details may be found in Wolter (1985). VPLX will generate Hadamard matrices for many of the standard sizes; further details are in Chapter 12.

Grouping Clusters. Half-sample replication is defined in terms of 2 selections per stratum, so that grouping or some other arrangement is required in other situations. If four clusters have been sampled from a stratum, choices include combining pairs of clusters into 2 superclusters, or dividing the stratum into 2 pseudo-strata of 2 clusters each. These strategies extend readily in instances with an even number of clusters per stratum. Strata with 3 clusters, which the stratified jackknife deals with happily, require painful compromise with half-sample replication.

How Many Replicates? Half-sample replication typically requires more calculation per replicate than the jackknife. Hence, except for situations in which highly precise variance estimates are required, the range of 25 to 200 represents a starting point for consideration, with the lower end representing relatively imprecise estimates.

Example. Under the same stratification as in the example for the stratified jackknife, the following VPLX run employs an input file in which the total sample and each half sample are represented by separate records on the incoming file:

```
comment  EXAM6


comment  This next example uses half-sample replication,
         but uses the same stratification as example EXAM5

create  in = example6.dat  out = example6.vpl

input    rooms persons cluster                                          #1
```

```
      3 variables are specified

replicate number  cluster

replicate count   5

format   (3f2.0)

replication  method   half-sample                                          #2


comment   EXAMPLE6.DAT contains:                                          #3
 5 7 0
 6 8 0
 5 2 0
 4 1 0
 8 4 0
 8 2 0
 5 7 1
 5 2 1
 8 4 1
 5 7 2
 4 1 2
 8 2 2
 6 8 3
 5 2 3
 8 2 3
 6 8 4
 4 1 4
 8 4 4

    Size of block   1  =              3

    Total size of tally matrix =      3

    End of primary input file after obs #      18


transform  in = example6.vpl out=exampl6a.vpl

user2

old  rooms persons

derived   proom

    (assigned to block   2)

labels  rooms 'Number of rooms' persons 'Persons'
        proom   'Rooms per person'
REPLICATE  0, V1=   36.00, V2=   24.00 RATIO=  1.5000                     #4
REPLICATE  1, V1=   36.00, V2=   26.00 RATIO=  1.3846
REPLICATE  2, V1=   34.00, V2=   20.00 RATIO=  1.7000
REPLICATE  3, V1=   38.00, V2=   24.00 RATIO=  1.5833
REPLICATE  4, V1=   36.00, V2=   26.00 RATIO=  1.3846


display  in = exampl6a.vpl
```

```
    Unnamed scratch file opened on unit 13

list     rooms  persons total (rooms persons) proom

cov      total (rooms  persons)

                                                  Estimate      Standard error

Number of rooms         :  MEAN                     6.0000              .2357

Persons                 :  MEAN                     4.0000              .4082

Number of rooms         :  TOTAL                   36.0000             1.4142

Persons                 :  TOTAL                   24.0000             2.4495

Rooms per person        :  VALUE                    1.5000              .1356

                      Covariances of the Sample Estimates

                                    Estimate              1              2


 Number of rooms         :  TOTAL
1                                    36.0000   .20000000D+01

 Persons                 :  TOTAL
2                                    24.0000   .20000000D+01   .60000000D+01
```

Exhibit 2.8. Annotated output from the first half-sample example.

Starting with the INPUT statement at #1, the specification of the problem is modestly different from previous examples. Two statements, REPLICATE NUMBER, which indicates that the variable `cluster` is not intended for a stratified jackknife but used instead to identify replicates on the file; and REPLICATE COUNT, which gives the total number of replicates, including the original sample; make their first appearance here. A REPLICATION METHOD statement at #2 is also included in order to direct VPLX to use the usual form of half-sample replication, that is, in which each replicate represents approximately one half of the overall total.

The comment at #3 lists the input data. Six records represent the overall sample, with `cluster` set to 0. Each of four half samples is composed of three observations. Orthogonality is achieved since each cluster is in exactly one half of the half samples (in this case, 2) and each pair of clusters in different strata are in exactly one quarter of the half samples (in this case, 1). (These two conditions are sufficient, but not necessary, to assure orthogonality.)

The estimates of totals shown at #4 were formed by multiplying the half sample estimates by 2; for example, 36.0 = 2*(5+5+8), for the first half-sample value for rooms. The variance estimator used by VPLX is:

$$Var_{hs}(X_0) = \frac{1}{n}\sum_{i=1}^{n}(X_r - X_0)^2, \qquad (2.4)$$

where *n* denotes the total number of half-sample replicates. Note that the estimated standard error for proom, .1356, is somewhat different from the results of linearization, .1284, and the stratified jackknife, .1297. All other estimated standard errors and covariances are identical.

Half-sample replication appears to enjoy some advantages over both the jackknife and linearization with respect to some nonsmooth statistics, such as percentiles.

Statistically Equivalent Representations. The input, EXAMPLE6.DAT , had shorter records than EXAMPLE5.DA T but also considerably more records, 18 instead of 6. Other methods of expressing the data may be substantially more efficient on occasion. For example:

```
comment  EXAM7


comment  This next example uses half-sample replication,
         but uses the same stratification as example EXAM4

create  in = example7.dat  out = example7.vpl

input    rooms persons weight repw1 - repw4                                  #1

      7 variables are specified

format   (2f2.0,4x,f2.0,4f2.0)

replication  method  reweighted half-sample


comment   EXAMPLE7.DAT contains:                                             #2

 5 7 1 1 1 2 2 0 0
 6 8 2 1 1 0 0 2 2
 5 2 3 2 1 2 0 2 0
 4 1 4 2 1 0 2 0 2
 8 4 5 3 1 2 0 0 2
 8 2 6 3 1 0 2 2 0


          Note how the half-samples are indicated by the replicate
          weights, for example, the first half-sample is formed by
          weighting observations 1, 3, and 5, by 2.

    Size of block   1  =             3
```

```
    Total size of tally matrix =      3

    End of primary input file after obs #       6


transform  in = example7.vpl out=exampl7a.vpl

user2

old  rooms persons

derived   proom

    (assigned to block   2)

labels  rooms 'Number of rooms' persons 'Persons'
        proom   'Rooms per person'
REPLICATE  0, V1=   36.00, V2=   24.00 RATIO=  1.5000                          #3
REPLICATE  1, V1=   36.00, V2=   26.00 RATIO=  1.3846
REPLICATE  2, V1=   34.00, V2=   20.00 RATIO=  1.7000
REPLICATE  3, V1=   38.00, V2=   24.00 RATIO=  1.5833
REPLICATE  4, V1=   36.00, V2=   26.00 RATIO=  1.3846


display  in = exampl7a.vpl

    Unnamed scratch file opened on unit 13

list     rooms  persons total (rooms persons) proom

cov      total (rooms  persons)

                                                Estimate      Standard error

Number of rooms         :  MEAN                   6.0000               .2357

Persons                 :  MEAN                   4.0000               .4082

Number of rooms         :  TOTAL                 36.0000             1.4142

Persons                 :  TOTAL                 24.0000             2.4495

Rooms per person        :  VALUE                  1.5000               .1356

                     Covariances of the Sample Estimates

                                     Estimate              1              2


    Number of rooms         :  TOTAL
   1                                  36.0000   .20000000D+01

    Persons                 :  TOTAL
   2                                  24.0000   .20000000D+01   .60000000D+01
```

Exhibit 2.9. Annotated output from a second example of half sample replication, using a different form of input.

The INPUT statement at #1 includes variables with reserved meanings: `weight` and `repw1-repw4`, that is, `repw1`, `repw2`, `repw3`, and `repw4`. The values of `weight` are all 1; this is the weight used to form estimates for the overall sample. Each half-sample replicate is derived by weighting the data by `repw1`, `repw2`, `repw3`, or `repw4`, respectively. Note that the observations included in half-sample 1 in `EXAMPLE6.DAT` each receive a weight of 2 in `EXAMPLE7.DAT`, and similarly for the rest of the replicates. The REPLICATION METHOD statement specifies the reweighted half-sample method, which is reweighted in the sense that `repw1`, etc., already incorporate the factor of 2 used in the first half-sample example. There are no differences in estimates between Exhibits 2.8 and 2.9; the only issue in the comparison is the different possible expressions of the replication method in the construction of the data set.

2.6 Generalized Replication

VPLX permits the calculation of the variance estimator:

$$Var_g(X_g) = \sum_{r=1}^{n} b_r (X_r - X_0)^2, \qquad (2.5)$$

where n represents the total number of replicates and b_r, $r=1,...,n$, are a set of coefficients supplied by the user. Similarly, the construction of the X_r must also be specified through a means such as `repw1`,.... For illustration, the previous example may be run as a special case of generalized replication:

```
comment  EXAM8


comment  This example is statistically identical to
         EXAM7, but illustrates the coefficient feature
         of VPLX giving the user flexibility in determining
         the replication calculation for generalized replication

create  in = example7.dat  out = example7.vpl

input    rooms persons         weight repw1 - repw4

      7 variables are specified

format   (2f2.0,4x,5f2.0)

coefficients   4 * .25                                                  #1

    Generalized replication assumed                                      #2

    Size of block   1  =             3

    Total size of tally matrix =     3
```

```
    End of primary input file after obs #       6

transform  in = example7.vpl out=exampl7a.vpl

user2

old  rooms persons

derived   proom

    (assigned to block   2)

labels  rooms 'Number of rooms' persons 'Persons'
        proom   'Rooms per person'
REPLICATE  0, V1=   36.00, V2=   24.00 RATIO=  1.5000                          #3
REPLICATE  1, V1=   36.00, V2=   26.00 RATIO=  1.3846
REPLICATE  2, V1=   34.00, V2=   20.00 RATIO=  1.7000
REPLICATE  3, V1=   38.00, V2=   24.00 RATIO=  1.5833
REPLICATE  4, V1=   36.00, V2=   26.00 RATIO=  1.3846


display  in = exampl7a.vpl

    Unnamed scratch file opened on unit 13

list     rooms  persons total ( rooms persons) proom

                                              Estimate      Standard error


Number of rooms         :  MEAN                 6.0000              .2357

Persons                 :  MEAN                 4.0000              .4082

Number of rooms         :  TOTAL               36.0000             1.4142

Persons                 :  TOTAL               24.0000             2.4495

Rooms per person        :  VALUE                1.5000              .1356
```

Exhibit 2.10. Annotated output showing half-sample replication implemented as a special case of generalized replication.

A COEFFICIENTS statement at #1 provides the values of b_r, which are in fact the values used by VPLX for half-sample replication. By providing this statement and replicate weights `repw1-repw4`, VPLX implements the generalized replication option, acknowledged at #2. The replicate values, at #3, are the same as in Exhibits 2.8 and 2.9.

Modified Half-Sample Replication. As a second example, it is possible to replace the weights of 0 and 2 used in half sample replication with modified weights of 0.5 and 1.5. For estimates of total, the new replicate represents a simple average of the half-sample replicate with the result

for the overall sample. Because each replicate consequently differs from the overall sample by half the distance of half-sample replication, the appropriate coefficients are 4 times those for the equivalent half-sample problem.

```
comment  EXAM9


comment  This example is similar to EXAM8, although now the
         replicate weights have been changed from 2 and 0
         to 1.5 and .5.  Note the corresponding change in
         the coefficients statement

create  in = example9.dat  out = example9.vpl

input    rooms persons         weight repw1 - repw4

      7 variables are specified

format   (2f2.0,4x,f2.0,4f4.0)

coefficients   4 * 1.0                                                  #1


comment   The contents of EXAMPLE9.DAT:                                 #2

 5 7 1 1 1 1.5 1.5 0.5 0.5
 6 8 2 1 1 0.5 0.5 1.5 1.5
 5 2 3 2 1 1.5 0.5 1.5 0.5
 4 1 4 2 1 0.5 1.5 0.5 1.5
 8 4 5 3 1 1.5 0.5 0.5 1.5
 8 2 6 3 1 0.5 1.5 1.5 0.5

   Note: compare to EXAMPLE7.DAT in EXAM7

    Generalized replication assumed

    Size of block   1  =              3

    Total size of tally matrix =      3

    End of primary input file after obs #       6

transform  in = example9.vpl out=exampl9a.vpl

user2

old  rooms persons

derived   proom

    (assigned to block   2)

labels  rooms 'Number of rooms' persons 'Persons'
        proom   'Rooms per person'
REPLICATE  0, V1=   36.00, V2=   24.00 RATIO=  1.5000                   #3
REPLICATE  1, V1=   36.00, V2=   25.00 RATIO=  1.4400
REPLICATE  2, V1=   35.00, V2=   22.00 RATIO=  1.5909
```

```
REPLICATE  3, V1=   37.00, V2=   24.00 RATIO=  1.5417
REPLICATE  4, V1=   36.00, V2=   25.00 RATIO=  1.4400


display  in = exampl9a.vpl

    Unnamed scratch file opened on unit 13

list     rooms  persons total(rooms persons) proom

cov      total(rooms  persons)

                                                 Estimate      Standard error

Number of rooms         :  MEAN                    6.0000               .2357

Persons                 :  MEAN                    4.0000               .4082

Number of rooms         :  TOTAL                  36.0000              1.4142

Persons                 :  TOTAL                  24.0000              2.4495

Rooms per person        :  VALUE                   1.5000               .1312

             Covariances of the Sample Estimates

                                    Estimate              1               2


    Number of rooms         :  TOTAL
   1                                 36.0000   .20000000D+01

    Persons                 :  TOTAL
   2                                 24.0000   .20000000D+01   .60000000D+01
```

Exhibit 2.11. Annotated output from second generalized replication example.

The COEFFICIENTS statement at #1 reflects the appropriate values to use with the revised replicates. The data set described at #2 can be compared to #2 in Exhibit 2.9; in each case replicate weights of 0 have become 0.5, and those of 2 have become 1.5. This modification produces replicates more similar to the original values than for half-sample replication, as can be seen by comparing values at #3 of Exhibit 2.10 with #3 here. The estimated standard error for proom, .1312, is closer to the results for linearization, .1284, and the stratified jackknife, .1297, than the half-sample result, .1356. In general, modified half-sample replication represents a form of compromise between the jackknife and the usual half-sample replication, combining properties of both.

EXAM8 uses generalized replication to obtain the same numerical results as EXAM7; in fact, the results of any of the previous examples may also be obtained by construction of appropriate data sets and specification of the generalized replication option. Generalized replication plays a special role in reweighting, in Chapter 9.

2.7 A Brief Tour of a VPLX File

The previous sections illustrated ways in which VPLX files could be produced in a CREATE step, altered in a TRANSFORM step, or provided to DISPLAY in order to estimate variances, without, however, showing what any of the VPLX files contained. In fact, VPLX files are FORTRAN unformatted files and are not easily displayed directly. VPLX creates and reads these files with the same FORTRAN statements in every computer environment, but FORTRAN implementation of unformatted files may vary. Generally, unformatted files cannot be directly copied across different computer systems. An EXPORT and IMPORT feature of VPLX provides a translation into and back out of a character representation for movement across systems, should the need arise. (EXPORT and IMPORT use a FORTRAN D24.16 format to preserve essentially all of the numerical precision of the original file, so the character representation of a VPLX file is about 3 times the size of the original unformatted version.)

EXPORT also has a didactic function, to present a simple listing of the contents of a VPLX file to illustrate the information provided on such files. (The listing shows estimates with reduced precision for ease of reading.) This section will display such a listing for a VPLX file in a previous example. This section is optional reading and is intended as an aid to the curious.

Example 5, shown in Exhibit 2.7, has been modified by replacing the DISPLAY step with an EXPORT step. Because no OUT= file is given, the step produces only the printed summary.

```
comment   EXAM5A


comment   The next example illustrates the stratified jackknife
          The six observations are grouped into three strata of
          two observations each.

create  in = example5.dat  out = example5.vpl

input    rooms persons cluster stratum

      4 variables are specified

format   (4f2.0)


comment   EXAMPLE5.DAT contains the following data:
 5 7 1 1
 6 8 2 1
```

```
 5 2 3 2
 4 1 4 2
 8 4 5 3
 8 2 6 3

    Stratified jackknife replication assumed

    Size of block   1  =              3

    Total size of tally matrix =      3

    Unnamed scratch file opened on unit 13

    Unnamed scratch file opened on unit 14

    Unnamed scratch file opened on unit 15

    End of primary input file after obs #       6

         3 strata observed on incoming file

transform  in = example5.vpl out=exampl5a.vpl

user2

old  rooms persons

derived   proom

    (assigned to block   2)

labels  rooms 'Number of rooms' persons 'Persons'
        proom  'Rooms per person'
REPLICATE  0, V1=   36.00, V2=   24.00 RATIO=  1.5000
REPLICATE  1, V1=   37.00, V2=   25.00 RATIO=  1.4800
REPLICATE  2, V1=   35.00, V2=   23.00 RATIO=  1.5217
REPLICATE  3, V1=   35.00, V2=   23.00 RATIO=  1.5217
REPLICATE  4, V1=   37.00, V2=   25.00 RATIO=  1.4800
REPLICATE  5, V1=   36.00, V2=   22.00 RATIO=  1.6364
REPLICATE  6, V1=   36.00, V2=   26.00 RATIO=  1.3846

export   in = exampl5a.vpl


IVERSN                                                                  #1
   9203

  NVTOT  NVREG NCLASS NVARID    NBY   NWGT   TYPE VROPTN NIDTOT  TSIZE NCLBLK
      5      3      0      2      0      0     18      4      5      4      2
 NCLBAR
      0

 BLTYPE BLXSTR BLXINC BLXSIZ BLVSTR BLVSIZ BLNCLS BLCPNT
      1      1      3      3      1      2      0      1

      0      4      1      1      3      1      0      1

Variable types
      1      1     11      1      1
```

```
Variable sizes
      1      1      1      1      1

Variable locations
      2      3      4

Crossed dimensions
      0

Variable names and labels
rooms        Number of rooms
persons      Persons
proom        Rooms per person
stratum      stratum
cluster      cluster

Level labels
      0

Variable names in crossings
      0

Number of replicates
      6

Coefficients
  .500   .500   .500   .500   .500   .500

   -9878.0000    -9878.0000        1.0000        1.0000        1.0000                  #2

       6.0000       36.0000       24.0000        1.5000                                #3

       1.0000        1.0000        1.0000        2.0000         .0000                  #4

       6.0000       37.0000       25.0000        1.4800

       1.0000        2.0000        1.0000        2.0000         .0000

       6.0000       35.0000       23.0000        1.5217

       2.0000        3.0000        1.0000        2.0000         .0000

       6.0000       35.0000       23.0000        1.5217

       2.0000        4.0000        1.0000        2.0000         .0000

       6.0000       37.0000       25.0000        1.4800

       3.0000        5.0000        1.0000        2.0000         .0000

       6.0000       36.0000       22.0000        1.6364

       3.0000        6.0000        1.0000        2.0000         .0000

       6.0000       36.0000       26.0000        1.3846
```

Exhibit 2.12. EXAM5A, including a listing from EXPORT.

Each VPLX file contains an initial record with a date indicating the last change to the design of the VPLX file, which is reported at #1. In this case, the last such change was March, 1992. The date of the file design usually differs from the program version indicated at the beginning of the output listing (as in Exhibit 2.2, although this first line was omitted from all other exhibits); in general, new features are often added to VPLX without any change to the design of the VPLX file.

The next record, with the associated FORTRAN variable names internal to the program, informs us that there are 5 variables overall: 3 are "regular," none are class, and 2 are variance identifiers. There are no "BY" groups (Chapter 7), and no weight. TYPE contains a code, 18, indicating how the replicates are to be defined, VROPTN 5 signals the stratified jackknife. Each replicate identification record contains 5 (NIDTOT) values. The total size of the matrix is 4 (TSIZE) cells per replicate, in 2 (NCLBLK) blocks with no (NCLBAR) associated block-class records.

The next display summarizes information for each block. The first block includes a cell for the estimated N (BLTYPE=1). The block starts in cell 1 (BLXSTR) of the overall matrix, with an increment of 3 (BLXINC) for each level of the class variables (had they been present) for a total data size of 3 (BLXSIZ). Similarly, the block begins with the first variable (BLVSTR=1), includes 2 (BLVSIZ) variables, has no associated classes. There is an assigned value of 1 to a pointer in the class block array, used only when classes are associated with the block. Information for the second block is recorded similarly; for example, no weighted N (BLTYPE=0) is present for this block.

The next information codes the types of each variable, which are real except for the single derived variable. Each variable has total size 1. A map to the starting locations of each variable follows next. There are no crossed dimensions in this example, but they would have appeared here, had they been present.

Variable names and associated labels appear next. Since there were no categorical variables, no level labels appear. Similarly, there are no variables involved in crossings. The VPLX file next records the number, 6, of replicates for variance estimation and the coefficients to be used in computing variance and covariance estimates.

The record at #2 reports that the first estimates are for the full sample. The record at #3 contains these 4 values, beginning with the estimated N. In general, TSIZE estimates for the full sample appear at this point. The record at #4 signals the construction of the first replicate, corresponding to `stratum` 1, `cluster` 1. The next three values encode the following instructions for constructing the replicate sample: observations in other strata are to be multiplied by 1, those in this strata but another cluster by 2, and those in this cluster by 0. (In this way, a VPLX file records how replicates have been constructed, and this information is used by REWEIGHT.)

This record is followed by the replicate values for the first replicate. The balance of the file contains similar information for the other replicates.

Other VPLX files appearing in examples in this chapter could be similarly displayed. Each would be similar in overall design but reflect differences according to the number of variables, choice of replication method, and so forth.

The listing from EXPORT provides an indication of how the VPLX file is arranged, but it is not as useful a diagnostic tool as other features available in VPLX. The CONTENTS step provides a clearer summary of variable attributes and other characteristics. A REPPRINT feature in the TRANFORM step is available to print replicate estimates for any specified set of variables.

2.8 For Further Reading

Cochran (1977) continues as a basic introduction to the theory of finite population sampling, particularly from the design-based perspective. Kalton (1983) and Lee, Forthofer, and Lorimor (1989) are among several somewhat less technical introductions. There are other perspectives on the theory of estimation from complex samples, including Särndal, Swensson, and Wretman (1992), and Little (1991).

Rao and Bellhouse (1988, 1990) prepared a broad summary of the history of inference from sample surveys. Papers representing milestones in the development of the theory include Binder (1983) for generalization of the linearization method to a broad class of estimators, Jones (1974) for work on the stratified jackknife, Kish and Frankel (1970, 1974) for their comparisons of variance estimation procedures for complex statistics, and McCarthy (1969) for half-sample replication.

Wolter (1985) serves as a general overview of design-based variance estimation for complex samples, with chapters 2, 3, and 4 covering the random group, half-sample, and jackknife methods, respectively. Efron (1982) presents a general perspective on replication methods. Other references on variance estimation for complex samples include Rust (1985), Krewski and Rao (1981), Kovar, Rao, and Wu (1988), and Rao and Wu (1988). Rao and Wu (1988) present important work on the stratified jackknife, providing some balance to the somewhat cautionary remarks on this procedure found in Wolter (1985) and Särndal, Swensson, and Wretman (1992). The modified half-sample replication described in Section 2.6 has been discussed by Fay (1989) and Judkins (1990).

Replication methods may have a critical role to play in the analysis of surveys with missing data, as Rao and Shao (1992), Fay (1991) and others extend and develop alternatives to the ideas of Rubin (1987), which also incorporated aspects of replication.