THE DEVELOPMENT OF AUTOMATIC CALCULATION OF THE COEFFICIENT OF VARIATION WITHIN THE GENERALIZED TABULATION SYSTEM

Victor A. Baxter, Internal Revenue Service

This paper describes the genesis of a program to automatically calculate coefficients of variation (C.V.) within the Census Bureau's generalized tabulation system (GTS). Some background is given first on the history of C.V. calculations at the IRS Data Center. The modifications made to GTS are described next, followed by an analysis of test results on typical applications. Throughout the paper, it is implicitly assumed that C.V.'s are being calculated from stratified random samples like those employed in the IRS Statistics of Income series.

1. BACKGROUND

Data processing history records an era during which programming was accomplished using machine level code. This required laborious coding of a seemingly endless string of binary values to obtain even the most trivial computer product. Eventually assembler languages were developed in which mnemonic symbols and decimal values replaced laborious binary coding. As a result, the work became considerably easier for the computer programmer.

Sometime later, higher level languages were written in assembler code, and with a single command a programmer could invoke a series of assembler level instructions. At about the same time, the generalized programs began to appear. The most famous and useful of these was the IBM generalized Sort/Merge Utility program. The Sort allowed the user to specify a few parameters and invoke a whole program of thousands of assembler level commands. Finally, generalized systems began to appear in which user parameters could invoke a series of programs. The Generalized Table producing System (GTS), developed at the Bureau of the Census [1], is an example of such a vehicle. The ability to choose between higher level languages or a generalized system made it possible to run many tabulations of large data files in a reasonable period of time (a staff year as opposed to the previous life-time or two).

In order to compile statistics on the characteristics of Individual Income taxpayers, for instance, we find ourselves involved with hundreds of millions of computer records residing on thousands of reels of magnetic tape. Such files are incontrovertably described as very large files. Even the most sophisticated techniques on the most powerful computers will require several days to process such a file even once. A requirement to produce several hundred tables using a very large file could require years of computer support. Also, when working with very large files, some severe operational problems can arise. Included among them is the clerical burden of keeping track of all the magnetic tapes involved. Some of the modern operating systems restrict the user to as few as 256 reels of tape at one time. In order to overcome these and other problems related to the immense volume problem, IRS typically tabulates just samples of the very large files.

Coefficient of Variation

Almost from the beginning of the Statistics of Income (SOI) program (and, of course, well before the age of computers), sampling was employed. Information on the variability introduced by sampling first began to appear in the published SOI volumes in the 1940's. Typically, in IRS sampling applications, the coefficient of variation has been the measure of sampling variability used.

The standard definition of a coefficient of variation (CV) can be written in percent as

$$CV = (100)(S / X)$$

where S is the standard error of the estimate X.

Range Method.-- When statistical data processing first started at the Data Center in 1965 the range [2] was used to calculate coefficients of variation. Of course, by employing the range it is not necessary to estimate S by the conventional sum of squares method. (This was an era at the IRS Data Center when people were available who understood both taxes and computer programming. It seem too much to expect to ask these people to deal with the second moment about the mean as well. Besides, COBOL and FORTRAN were new languages to IRS programmers that were not accepted quite yet, so everything was done at the assembler level, or sometimes at the actual machine level. In any case, justifying an easily described calculation involving the range seemed to be very natural.)

Let R be the range of n observations, i.e., the difference between the highest and lowest of the n values. The range estimator of the standard deviation is

$$\hat{S} = R / D,$$

where D is a constant that depends on sample size. (For values of n between 3 and 10 D is approximately equal to the square root of n.) In the case of n = 5, the quantity 1/D = 0.4299 so the CV is estimated by

$$\hat{CV} = (42.99) (R / X)$$

The technique, as applied then, consisted of randomly assigning numbers one through five to the records in the file of tax returns [3], thus creating a replicated file with five replicates. When the data were tabulated, five copies of the table matrix were created - one for each replicate. This was followed by a summarizer program which read in five matrices and added them together to obtain a final cell estimate. For each cell, the largest of the replicates (MAX) and the smallest (MIN) were selected; the difference (MAX)-(MIN) was multiplied by one hundred and divided by the cell estimate to yield the range as a percentage of the cell estimate.

This technique had several drawbacks. The data cells were accrued using weighted records and the five matrices received by the summarizer program reflected data summed over several thousand observations. Much information about variability was lost in this process. In spite of its shortcomings, though, the range method was used for several years. It was easy to program and worked well as a method of getting rough order of magnitude estimates for C.V.'s.

Later Methods.-- With the advent of third generation computers for statistical processing at the IRS, the time had come for a more sophisticated approach. At first the approach to the second moment about the mean was via an assembler level IBM S/360 program using packed decimal arithmetic. Of necessity, heavy emphasis was laid on maintaining the proper number of decimal places at each intermediate step to insure validity of the final figure. Later, the calculation program was rewritten to use floating point arithmetic and the precision problem disappeared.

In keeping with the spirit of protecting the business-oriented computer programmer from the details of statistical theory, a coefficient of variation calculation utility program was written. The user of the C.V. program was required to furnish a weight file and an unweighted stub record file and received as output a C.V. file. This same technique was used for both frequencies and amounts.

The weight file contained one record for each weight in the sample. Each record contained a series of codes, to allow association of the cell records with the weights, a sample size, and a variance factor. The variance factor (VF) was of the form:

$$VF = (W)(W-1)$$

Internally, the C.V. program determined the Weight (W) by using the relationship:

$$W = (SQRT(4*VF + 1) + 1) / 2$$

where SQRT is a square root function. Note that the trivial case VF = 0 yields W = 1.

The square root algorithm used in the C.V. program was the classical Newton-Raphson method [4] using:

$$SQRTC = (SQRTP + ARG / SQRTP) / 2$$

where SQRTC is the current approximation of the square root, and ARG is the input argument.

The input argument was in floating point. For an initial approximation, an input argument of zero or one was returned without further inspection. In the case of an input argument less than one, the mantissa was doubled and the characteristic was halved to yield the initial approximation. In the case of an input argument greater than one, the initial approximation was obtained by halving the mantissa and halving the excess, over hexadecimal 41, of the characteristic. The iterations were terminated if:

$$ARG / SQRTP = SQRTP$$

or until a preset number of iterations had been carried out.

The weights were maintained in an internal table for later use as the stub file records were processed. The stub file records contained a sample code, to allow association with the weight codes, and the data values falling into a stub from a single observation. This technique shielded the user from the necessity of obtaining sums of squares, but it caused the generation of an enormous number of records as input to the C.V. calculation program. It was often the case that several million records were processed by the C.V. calculation program.

On some occasions the most difficult problem facing the user of the C.V. calculation program was the operational difficulty caused by so many records. Another problem that arose was one of table specifications. Generally, a team of programmers would write a table producing program and at some later time another team would write a series of programs to produce that same table with coefficient of variation figures. Under these conditions there would often be an error in the C.V. tables that was due to a lack of understanding of the table specifications rather than an error in the C.V. calculation program. After this C.V. program had been in use for a few years, IBM equipment was replaced with UNIVAC and a new approach was necessitated.

2. ALTERED C.V. APPROACH

There were literally thousands of programs that had to be converted from IBM to UNIVAC so a simplified approach to the problem of writing table producing programs and calculating coefficients of variation seemed to be in order. In the face of the enormous resource drain involved in the program conversion effort, the writing of a generalized table producing system in-house was unthinkable. A search of vendor software and UNIVAC oriented government agencies yielded a generalized table producing system written by the Bureau of Census, called GTS. The GTS package was rated as excellent by Ivor Francis [5] in tabulating power and language simplicity when compared with other packages (i.e., OSIRIS, SPSS, TPL); hence it was chosen.

GTS offered the advantage of already having been written and tested for the UNIVAC. However, GTS came with its own set of disadvantages. The Census people usually dealt with a large number of small tables (less than 100 columns per table) whereas IRS normally requires larger tables. No coefficient of variation feature was present in GTS. GTS also employed a unique Input/Output (I/O) system called CENIO for its work units.

CENIO proved to be the worst initial problem. The Census people did not wish to provide a copy of the CENIO package for several reasons. One reason was that it had been developed over several generations of UNIVAC software and was now scattered over system libraries and required modifications to system utilities and compilers. The job of copying all those modules would have demanded extensive additions of user written code to the UNIVAC Operating System.

A second reason was data security. Many of the Census files used CENIO and because no one else had such an I/O package, this afforded those files a measure of data security. If IRS had been given CENIO, that security would have been compromised.

The suggestion from Census was to convert the calls to the CENIO package to conventional COBOL reads and writes. This worked well for the first few modules that were modified, but it was found that CENIO was being used in a manner not permitted by COBOL--the length field of variable length records was being passed back to the COBOL program. There was no way to duplicate this in a COBOL I/O structure. Things were at an impasse since the length count was not available from COBOL. Two basic approaches to a solution existed. One, alter the existing work records to include a length field, or two, write a unique assembler level I/O package to replace the missing CENIO feature. Logic would seem to be with the record alteration approach, yet changing the record would have required alteration of virtually every module in the GTS system. Thus, writing an assembler level I/O package was undertaken. This proved to be an extremely difficult task.

The programmer was required to respond to the I/O STATUS codes, and to build, read, and write the header and trailer labels. The programmer was also responsible for buffer allocation, control block generation, and keeping track of the current sector on disk files. There were no macros for open, close, read, or write functions. These problems were overcome and an imitation of CENIO was installed.

CV Feature

The next step was to be a coefficient of variation routine functioning in GTS. The goal was to shield the unsophisticated user from the statistical details and avoid the mistakes of past versions of C.V. modules. Two levels of command were chosen for the user interface. The first level was a "RUN CV" command to allow the user to describe the attributes of the sample design (sample sizes vs. population), the desired sigma level, and the print format desired (C.V.'s represented as percentages, or as alphabetic codes).

At the second level, since the user can request up to 100 distinct tables, at each table description, a "CV" or "NOCV" field can be added to the table description to indicate whether or not C.V. is desired.

Next came the problem of where to extract the data necessary to perform the computations. There were commands for column arithmetic, line arithmetic, table arithmetic and rolling of totals through levels of tables. It was decided to use the "TALLY" statement for the data source. When the "TALLY" statement was executed, in addition to accruing data into a table cell, transparently to the user, a sum of squares and sum cell was also accumulated. By accumulating a table image rather than stub records, and summarizing over records having a common sample size and population, the number of intermediate records was greatly reduced. Also, the standard record compression technique available in GTS via CENIO was used to further reduce the file space required for C.V. work files.

The rolling of table totals was still a problem. In order to capture the data needed for C.V. computations, the C.V. work files had to be summarized in the same manner as the tabulated data was summarized. The solution involved two modules. A C.V. calculation module was written for tables not requiring total tables to be constructed from detail tables and a more complicated C.V. module was written to support capturing of the sum of squares and squares of sum at each level of summarization. (See Appendix A for additional details.)

3. TESTING AND EVALUATION

One of the applications for GTS was a series of 28 tasks to be produced from a common file. It was decided to use this "live" application as a test vehicle for the coefficient of variation.

By removing the C.V. commands from a GTS application, a copy of the run for non-C.V. processing was created for comparing C.V. and non-C.V. runs.

The test vehicles for all trial runs (see figures 1 and 2) used an input data file of 66 records. In each test, 28 tables consisting of 39 rows and 16 columns were accumulated. For the purposes of the following memory allocation calculations, it should be noted that a single precision table cell requires one word and a double precision table cell requires two words of main memory on the UNIVAC 1100 systems.

Double precision arithmetic was requested necessitating 1248 words per table:

(39 rows X 16 columns X 2 words per cell
(for double precision)) = 1,248 words

For the non C.V. tables, a table accrual area of 34,944 words was required.

1,248 words per table X 28 tables = 34,944 words

For the C.V. runs, three accrual areas were used; one for the weighted sums, one for

Figure 1.--Trial Run Times for Test Data of GTS Programming Using the C.V. Feature (Run on UNIVAC 1182 at IRS Data Center)

Run Number	CPU Time	I/O Time	CC/ER Time	Wait Time	Elapsed Wall Clock Time
Total	18' 18.941"	55' 7.006"	24' 31.102"	39.567"	5 Hr. 32'22"
1	2' 14.573"	6' 53.484"	3' 3.914"	6.857"	1 Hr. 8'31"
2	2' 17.386"	6' 45.663"	3' 59.714"	3.811"	1 Hr. 3'40"
3	2' 35.341"	7' 47.183"	3' 5.947"	3.981"	0 Hr. 34'20"
4	2' 14.146"	6' 42.833"	3' 4.433"	6.202"	0 Hr. 59'13"
5	2' 14.267"	6' 43.536"	3' 4.552"	4.816"	0 Hr. 25'56"
6	2' 14.264"	6' 44.135"	3' 4.363"	3.496"	0 Hr. 18'38"
7	2' 14.360"	6' 44.910"	3' 4.725"	5.787"	0 Hr. 22'55"
8	2' 14.610"	6' 45.261"	3' 4.327"	4.617"	0 Hr. 39'09"
Average	2' 17.368"	6' 53.376"	3' 3.888"	4.946"	0 Hr. 41'32.75"

NOTE: ' = minutes; " = seconds.

Figure 2.--Trial Run Times for Test Data of GTS Programming Not Using the C.V. Feature (Run on the UNIVAC 1182 at IRS Data Center)

Run Number	CPU Time	I/O Time	CC/ER Time	Wait Time	Elapsed Wall Clock Time
Total	9' 28.392"	33' 23.586"	15' 24.811"	55.240"	3 Hr. 46'23"
1	1' 10.394"	4' 1.327"	1' 51.478"	3.261"	0 Hr. 45'57"
2	1' 11.953"	4' 28.592"	1' 53.974"	3.386"	0 Hr. 46'47"
3	1' 10.793"	4' 8.449"	1' 56.554"	3.316"	0 Hr. 12'23"
4	1' 10.879"	4' 8.717"	1' 56.638"	25.026"	0 Hr. 49'15"
5	1' 11.134"	4' 8.807"	1' 56.416"	3.216"	0 Hr. 15' 3"
6	1' 11.044"	4' 8.811"	1' 56.435"	6.881"	0 Hr. 21'15"
7	1' 10.981"	4' 9.264"	1' 56.782"	4.752"	0 Hr. 16'22"
8	1' 11.128"	4' 9.619"	1' 56.534"	5.402"	0 Hr. 19'21"
Average	1' 10.049"	4' 10.448"	1' 55.601"	6.905"	0 Hr. 28'18"

NOTE: ' = minutes; " = seconds.

unweighted sums, and the third for unweighted sums of squares. Thus, an accrual area of 104,832 words was required for the C.V. runs:

34,944 X 3 areas = 104,832 words

The C.V. runs generated a COBOL program of 6,984 statements and the non-C.V. runs generated a COBOL program of 3,240.

Results of Test

It is presumed that the bulk of the difference in central processing (CPU), input/output (I/O), and channel command and executive request (CC/ER) times between C.V. and non-C.V. runs is due to the increased workload on the COBOL compiler, and that a lesser portion of the difference is attributed to the additional accrual requirements in the tabulation module as well as the additional work required in the print module.

The wide variance in the elapsed wall clock times is attributed to the job mix on the computer. Runs that occur during prime time tend to run slowly due to the system degredation brought on by an attempt to simultaneously service many users. The runs started on the off-shifts required less wall clock time even though the CPU, I/O, and CC/ER times are about the same.

As a general observation, there is a measurable penalty to be paid for C.V. calculations. However, in view of the dominating effect of job mix on turnaround time, this penalty is greatly minimized.

4. CONCLUSION AND AREAS FOR FURTHER EVALUATION

The implementation of C.V. in GTS is a boon to IRS table producing efforts. For the first time there is a common module for C.V. calculation used by all of the IRS programmers. By simply invoking a single GTS command "RUN CV", GTS users can easily obtain C.V. data. Also, in the past, if a C.V. requirement had been overlooked during the systems analysis of a series of table producing programs, the attempt to provide for C.V. after the programs were written was either very expensive or impossible with the existing design framework. Now, with C.V. in GTS this problem can be remedied in a matter of a few minutes of programmer time.

For the near future, we plan on additional GTS enhancements to support the electronic composition of published tables, and new language elements to automate the elimination of disclosure problems for sensitive data; also planned are automated routines for the detection of excessive sampling variability.

ACKNOWLEDGEMENTS

The author would like to thank Ralph Bristol of the Office of Tax Analysis, Joe Higgins of the Internal Revenue Service, and Bruce During and Elbert Williams of the Bureau of the Census for their contributions to this paper. Special thanks is also extended to Susan Hinkins of the Internal Revenue Service for her assistance on preparing the Appendix. In addition, appreciation is extended to Bob Cripe and Florine Morrison for typing.

NOTES AND REFERENCES

[1] U.S. Bureau of the Census, Generalized Tabulation System, an unpublished users' manual.

[2] Natrella, Mary, Experimental Statistics, National Bureau of Standards Handbook 91, October, 1966 Edition, Chapter 2, p. 6.

[3] These assignments were made systematically within each sample strata separately.

[4] Pennington, Ralph, Computer Methods and Numerical Analysis, Second Edition, 1970, MacMillan.
In pages 286 through 292, the author develops Newton-Raphson via a Taylor Series approximation; Pennington's note on speed of convergence will be of interest to those who prefer rational functions.

[5] Francis, Ivor and Sedransk , Joseph, Comparing Software for Processing and Analyzing Survey Data, ISI Bulletin, 1979, p. 48.

The sample is divided into strata identified by sample codes h, $h = 1,2...H$. Each sample code is reviewed so that weights can be determined. In the remarks that follow, MODUSORT 1, MODUSORT 2, and TABOUT are three disk work files used for the storage of intermediate results.

The following notation applies to the units in stratum h. N_h is the number of units in the population for sample code h. n_h is the number of units in the sample for sample code h.

The specified tabulation defines characteristics which divide the population into subpopulations, or domains. For example, the sample may have been stratified on income level, but the desired tabulation is by occupation code. The tabulation defines the table cells j, $j = 1, 2...J$. Currently GTS only calculates the C.V. for the estimated cell total.

The following notation applies to the sample units in stratum h which lie in the subpopulation j. A_{hj} is the sum of the measurements from returns with sample code h which lie in cell j (i.e., the summation is over all returns which are in both stratum h and cell j). B_{hj} is the sum of the squares of the measurements from returns with sample code h which lie in cell j.

On a record by record basis, the value of each A_{hj} is calculated and stored on the MODUSORT1 file, and the value of each B_{hj} is calculated and stored on the MODUSORT2 file.

Each cell estimate is calculated, on a record by record basis also, and stored on the file TABOUT. This estimate, E_j, is defined by

$$E_j = \sum_h (N_h / n_h) A_{hj} .$$

After all records are processed, the MODUSORT1 and MODUSORT2 files are sorted by table name, table part, and sample code. The TABOUT file is sorted by table name and table part.

If higher level tables are to be created from the summarization of lower tables (for example, a U.S. TOTAL table is to be created by adding regional tables), a new record is created on each work file by summing the records across cells. For example, on MODUSORT1 a new record would be created containing the values of $\Sigma_j B_{hj}$ for each h.

After all higher tables are created, the C.V. for each cell j is calculated as follows. For each stratum h,

$$T_{hj} = (Z_h)(B_{hj} - (A_{hj})^2 / n_h)$$

is calculated where

$$Z_h = \begin{cases} (N_h (N_h - n_h))/(n_h (n_h - 1)), \\ \qquad \text{if } n_h > 1; \text{ and} \\ 0, \qquad \text{if } n_h = 1. \end{cases}$$

Note that T_{hj} is the component of variance from the hth stratum. If $Z_h = 0$, the cell value contribution comes from a single observation and the C.V. is not computed; an asterisk character (*) is printed instead. If each n_h is greater than 1, the standard deviation of the estimated total for cell j is

$$S_j = \sqrt{\sum_h T_{hj}} .$$

The C.V. is then computed as

$$(100)\ (S_j / E_j)$$

for each cell.

In the edit run, the C.V. figure is printed with three digits to the left of decimal and two digits to the right of the decimal. If the single observation situation should arise, two* characters are printed.

The user is allowed to provide a sigma-level for C.V. calculation in which case the final formula changes to:

$$V = 100\ (\text{sigma-level})\ S/E.$$

If no sigma-level is provided, a default of one is used.

All of the intermediate calculations are carried out using floating point arithmetic. The user decides on single precision or double precision on a table by table basis.