Paper 3049-2015

Survey Sample Designs for Auditing, Fraud, and Forensics

Turner Bond, Statistician, HUD - Office of Inspector General.

ABSTRACT

Sampling for audits and forensics presents special challenges: Each survey/sample item requires examination by a team of professionals, so sample size must be contained. Surveys involve estimating--not hypothesis testing. So “power” is not a helpful concept. Stratification and modeling is often required to keep sampling distributions from being skewed. A precision of alpha is not required to create a confidence interval of 1-alpha, but how small a sample is supportable? Many times replicated sampling is required to prove the applicability of the design. Given the robust, programming-oriented approach of SAS®, the random selection, stratification, and optimizing techniques built into SAS can be used to bring transparency and reliability to the sample design process. While a sample that is used in a published audit or as a measure of financial damages must endure a special scrutiny, it can be a rewarding process to design a sample whose performance you truly understand and which will stand up under a challenge.

INTRODUCTION

While a lot of attention has been paid recently to the use of analytics to detect fraud/waste/abuse, someone eventually has to confirm and quantify the fraud in order to make a case. This inevitably involves a sample of cases that must be reviewed by qualified professionals. Oftentimes, this involves one of the 1.8 million accountants and auditors (USA only) and is typically done in an audit framework of some kind. For our purposes here we will characterize the process of analyzing samples for this purpose as “audit sampling.” In this paper we characterize the fraud, waste, or abuse that an auditor might find under the generic term of “fraud,” or in terms of sample design, as the “amount in error.”

Audit sampling has some things that set it apart from other sampling fields, such as quality control, biological outcomes or population counts. The sampling units in fraud detection are often measured in dollars and unlike QC samples the amounts can vary widely. Most audit sample universes are skewed heavily to the right. Where a QC sample might be akin to measuring the diameters of grains of sand on the beach, an audit sample can sometimes be like measuring particles from a pail-full of sand mixed with some pebbles and a few rocks that are golf ball sized and larger. Whether or not one of those golf-balls is included in the sample can make a big difference in the outcome if the sample isn’t designed properly. Even more confounding is the fact that, in an audit context, not all sample items return a value. If a sample item has no fraud it is essentially a null and returns a value of zero, regardless of its size. Since not all sample items will show fraud, waste or abuse we end up mixing samples with a value, into a sea of zeros to project any means or standard errors.

Like much of statistics, sampling theory and practice is based on an assumption that the data is somewhat normally distributed. In many cases, the sampling distribution of a somewhat normally distributed dataset will be smoothed out further by the central limit theorem, and methods based on the assumption of normality can be made to work. Because of the two factors mentioned above, not only will the sample universe be skewed in a fraud detection setting, but the true sampling distribution itself will not be made normal.

With enough samples one can overcome many obstacles of course, but in the case of audit sampling each sampling unit involves detailed review by a qualified professional. It’s not difficult to survey the family size of 100 households. It can be quite time consuming, however, to have a team of professionals examine the documentation on 100 failed mortgages. So, the sample must not only be well designed, but it must be efficient and compact.

CLASSIC PRECISION-BASED ESTIMATES

Unlike sampling in biomedical experiments or other hypothesis testing, audit sampling is an estimating technique. Parameter values in the universe are inferred from those of the sample and counts and sums

for the universe are estimated based on these values. This means that the concept of "power" is not helpful in determining the size of the sample (Vickers, 2010):

> *If the main purpose of your study is estimation, then the sample size you'll need depends on how precise you'd like your estimate to be* [the width of your confidence interval]. … *If the main purpose of your study is hypothesis testing then the sample size you'll need depends on power* [the likelihood of accurately rejecting the null hypothesis].
>
> -- Andrew Vickers

In the case of probe-sampling or other exploratory samples, methods using power or alpha and beta error have been used -- in lieu of probability formulas -- but generally speaking the design of the audit sample is selected based on confidence intervals, precision and similar measures derived from classic functions involving a probability distribution.

So, in a simple world, this would mean we could use formulas such as the following to estimate the sample size required (Cochran, 1977; Wayne W. Daniel, 1983):

$$n_{SRS} = \frac{Nz^2s^2}{d^2(N-1) + z^2s^2}$$

Since audit sampling often involves a known, finite universe of values, the variances (s^2) are calculated as one might expect: the average of the squared distances from the mean dollar value of all sample items. With stratification, the above formula gets a bit more complex, but the underlying idea remains the same. In traditional practice, the area under both tails from the stated confidence interval (i.e. α) is multiplied time the mean and squared to get d^2 so the precision is treated as ± the area outside the stated confidence interval.

PROBLEMS WITH CLASSIC APPLICATION OF PRECISION-BASED METHODS

While this classic approach is effective in many forms of estimation, it doesn't apply well to auditing and fraud detection. This approach assumes a static, knowable, valuation of sample items. While we know the original value of the sample items in our universe, we cannot know the value of the fraud our team of professionals will find in an individual sample item. It could be the full amount, it could be a fraction or, many times, it could be none. So, in practice, the true variance found during the audit will inevitably be greater than the population variance used to design a sample under a classic application of precision-based methods. The true variance will be driven by two forces:

1. Variability of Amounts – as affected by the value of the fraud or non-conforming amount, often referred to as the amount in error
2. Error Probability – as expressed in the proportion of sample items that are found to contain an error; otherwise referred to as the error rate.

The combined effect of these two forces can have a complex effect on overall variance. Dollars may be detected in one of the influential sampling units, for instance … or they may not. Because probability phenomena involve factorial equations, there is no practical way to combine these two and infer the resulting variances or probability distributions using calculus and minimization.

A further drawback to the classic application of precision-based methods is that they are static in nature. They are based on a single set of numbers with a single computation of variance derived from the population. In reality the sample variance that will be found during an audit has a dynamic range of possibilities based on the size of error found in the samples and the range of error rates that might be found.

THE MISSING LINK: MODELING THE ERROR FIELD

The key to accurately designing for the range of possible audit findings is to add a step to the design process in which we model the types of error that are likely to be encountered in an audit. The goal here is to provide a means of testing the performance of various sample designs under the conditions they might encounter when applied in the field. We call this "modeling the error field." While we don't know which combination of error rates and dollar amounts will be found, we can recreate the possible ranges of number variation that might be found and we can later design our sample around a range of possibilities instead of just one.

The process of modeling the error field is broken into two steps based on the two forces mentioned above: a) error amounts and b) error rates. In this case, we use the term "error amounts" to connote an amount of fraud waste or abuse that might be uncovered by an audit team.

MODELING ERROR AMOUNTS

SAS® provides a robust programming language when it comes to modeling the amount of error. Most of the amounts you can imagine can be described in code though there are some limitations in some of the more arcane sampling distributions. Many times, a simple uniform random distribution is enough to describe the amount of dollars in a sampling unit which might be in error.

Simple Application of a Uniform Distribution

For instance in the sample below, we are modeling the amount of error we expect to find in a type of PPS sample called a monetary unit sample where we sample a single dollar. The dollar starts off with a value of $1. The amount of that dollar an audit team finds to be "in error" might be a fraction of that. In this case, we are expecting that occasionally – about 20 percent of the time – the error will be less than or equal to half the amount in the sample item. We expect the remaining errors we find to be equal to the full amount of the sample item.

Below is some simple DATA step code that can be used to generate and apply that kind of error:

```
data MUS_test_UNIVERSE (keep =  amount rec_id strata) ;
 call streaminit(7) ;
 strata = 1 ;
 do i = 1 to 1111111 by 1 ;
   occasional = rand('uniform') ;
   partial = rand('uniform')  * .5 ;
   taint = 1 ;
   if occasional le .2 then taint = partial ;
   amount = taint ;
   rec_id = put(i,z6.0)  ;
   output ;
 end ;
run ;
```

There are, of course, any number of ways to model the type of error we might expect to see. The point here is not to exactly describe something that really can't be known. The point is to emulate a reasonable facsimile of the range numbers one might encounter and to do so in a way that shows some awareness of what a reasonable worst-case scenario might be. Similar to the way in which the central limit theorem makes the sampling distribution more normal than the distribution of the original universe, an error amount that roughly mimics the reasonably expected range of numerical variance, is sufficient to test how well a given sample design will smooth out variability and provide a confidence interval that accurately describes the results. In the previous sample code we used the STREAMINIT function to set the seed for the RAND() function so that results are reproducible each time we run the test. The STREAMINIT function is used once inside each DATA step. Use of the RANUNI() function also works well. Both, by the way, will produce the same results across different machines and SAS versions. One might also deduct a uniformly distributed random fraction from the base value, or modify the range of variation based

on something in the data. In a script-based coding environment, there are any number of ways one can emulate the range of amounts that you intend to test against your sample design.

Applying Other Distributions

In some cases you may find, or you may believe or know, that the probability distribution of error amounts approximates a known mathematical curve. You can apply the HISTOGRAM function in proc UNIVARIATE to organize the data and have SAS do a fitted application of curves that are commonly used to emulate probability distributions. It's important to remember here that our goal is to improve on the classic approach of estimating variance from the original number set which has a weak relationship to the variance obtained from the numbers you will actually find. Visual inspection is sufficient to determine if a known curve provides a close enough fit. Reviewing a Q-Q plot and quantile estimates can also be helpful, particularly if you want to inspect tail conditions. Some of the indexes that UNIVARIATE prints out are not helpful in evaluating general fit. When there are a lot of data points a Kolmogov-Smirnov reading, for instance, will reject the mathematical fit to a normal distribution even though your distribution may conform to a bell curve better than most things you find in nature.

In the example below, we profiled the range of difference between the stated and the actual loan-to-value ratio (LTV) in a given group of home loans. The amount of difference is expressed as a percentage of the true LTV amount in order to standardize this data across various sizes of loans. We had information from 63 loans that had been reviewed during an earlier process. Our goal here is simply to get a sense of how far off stated LTV ratios might be, and to get a general sense of whether these types of errors are distributed uniformly across a range or if they show a central tendency. Knowing these simple things will greatly enhance how well our sample design performs across the reasonable range of possible findings.

The UNIVARIATE code below profiles our standardized data and shows how a fitted application of a standard curve fits the general distribution of our data:

```
proc univariate data= LTV_OFF (rename=(LTV_spread=pointest))   pctldef=1 ;
   var  pointest  ;
   format pointest percent7.1  ;
   label pointest = "LTV Mistatements" ;
       histogram pointest /
                     cframe = ligr
                     cfill  = blue
                     midpoints = .0 to 1.00 by .04
                     normal(color=gold  w=3 l=2)
                     weibull (l=1  color=red sigma=EST theta=est)
                     gamma (l=1  color=violet sigma=EST theta=est)
                     name='LTV_err';
                     inset n / header="Test Samples" pos = (78,90) ;
 run;
```

UNIVARIATE will generate parameter estimates as it computes the best fit for these typical curves and those can be used in the DATA step to model the error amounts we are designing for. Since the number and names of parameters produced by UNIVARIATE don't always appear to match those in the SAS functions offered for the DATA step, a little careful study is sometimes required to reproduce probability distributions in the DATA step in SAS. In the case above we found that the Weibull function is a good fit for our data. Since the Weibull distribution is easy to work with and the SAS function for a Weibull distribution offers a full compliment of parameters, we can use the parameter estimates from UNIVARIATE to reproduce a Weibull distribution in our data. In this manner, we can model a level of uncertainty in our data that equals, but doesn't exceed the types of uncertainty that our final sample design will have to be able to absorb.

Taking the parameters generated by the curve fitting in UNIVARIATE we can generate a random Weibull distribution of error percentagbes in LTV and use it in additional steps to model the possible extent of dollar losses. The code below illustrates the application of Weibull parameters to help us model the expected variability and size of the dollar amounts affected:

```
data SIMULATED_ERR_AMTS ;  set UNIVERSE ;

< more code >

  LTV_diff = 0 ;
  call streaminit(7) ;
  if rand('uniform') le &err_rate. then do  ;
    LTV_diff = rand('weibull',2.966899,0.279575) + -0.18335 ;
  end ;

  /*** compute what actual LTVs and values might be ****/
  actual_LTV = SFDW_ltv - LTV_diff ;
  act_value = cap_loan/actual_LTV ;


< more code >

run ;
```

This practice of modelling the amount of misuse that an audit team might encounter, would not be part of traditional precision-based sample design. While it's tempting to view the above exercise as a fine-tuning or a marginal improvement, it is more than that. When we blended in the range of variability noted in the aforementioned Weibull distribution, we found that we needed to increase our sample size by about fifty percent to provide a viable projection.

MODELING THE PROBABILITY OF ERROR

The probability of finding a given error is the second component to the sample variance and is an important part of modeling the error an audit team might encounter. As noted earlier, the probability of finding a certain number of samples in error involves a probability equation. These involve the heavy use of factorials, which don't blend well with the simpler process of computing s^2. There are numerous combinations possible when we consider whether a particular sampling unit happens to be a zero finding or contain an error, whether it is selected in the sample and what the overall rate of error is. Below is the typical formula for selecting a particular number in error giving a certain ratio of error P, in the universe at large. As can be seen, probability formulas involve a significant number of factorials and this is one of several you might accumulate to calculate selection events:

$$\Pr(a) = \frac{n!}{a!\,(n-a)!} P^a Q^{n-a}$$

It's true that we often use other curves to mimic the behavior of probability calculations, but when it comes to merging variance calculations with the probabilities of finding an error in a given sample record, the original formulas are important and computing the true curve of the sampling distribution becomes too complex. As stated earlier assuming a normal distribution really isn't sufficient in this case. Because the projected results of the sample will be used in the negotiation for a legal settlement or a report to congress, we need to be able to demonstrate that our stated confidence intervals not only conform to defensible practice, but reflect what we actually find in nature.

Fortunately, we don't need to jointly compute the error probability and random error amounts to construct the sampling distribution of our sample. The probability of error can be treated as a separate layer and applied independently to the previously constructed range of amounts. The SURVEYSELECT procedure that SAS provides for randomly selecting samples is useful for constructing a range of error rates. Below is a simplified version of one method that can be used to randomly select error amounts at a specified rate of error.

In the example below the error rate has been set to 30 percent:

```
/*** (1) SET THE MODELED RATE OF ERROR OR FRAUD.  ***/
%let failrate = 30 ;
proc sort data=MODL_POP ; by strata ; run ;

/*** (2) MAKE A COPY OF THE POPULATION WITH NO ERRORS. ******/
data POPraw (keep=rec_id strata aud_diff fail ) ;
   set MODL_POP ;
  format aud_diff dollar12.2 ;
  fail = 0 ;
  aud_diff = 0 ;
run ;

/*** (3) MAKE A COPY OF THE POPULATION THAT IS 100% ERRORS. ***/
data ERROR_POOL(keep=rec_id strata aud_diff fail )
     SAMPRATE(keep = _rate_ strata) ;
  set MODL_POP(keep=rec_id amount strata) end=last ;
    by strata ;
  format aud_diff dollar14.2 ;

  fail = 1 ;
  aud_diff = amount ;
  output ERROR_POOL ;

  if last.strata then do ;
   _rate_ = &failrate/100 ;
  output SAMPRATE ;
  end ;
run ;

/**** (4) RANDOMLY SELECT THE SPECIFIED PERCENTAGE OF ERRORS. ****/
proc surveyselect noprint
  data= ERROR_POOL
  seed = 7
  samprate= SAMPRATE
  out = POPfail(drop=SamplingWeight SelectionProb)  ;
  strata strata ;
run ;

/**** (5) OVERLAY THE SELECTED ERRORS ON TOP OF THE POPULATION AT LARGE.
      SAVE AS A SEPARATE DATASET (POP_err30) TO TEST THE PERFORMANCE
      OF THE SAMPLE UNDER A 30 PERCENT RATE OF ERROR.  ****/

%SORT_N_MERGE(POPraw, POPfail, POP_err&failrate., sort_IDX, a ) ;
```

Simple macro code such as the following can be used to cycle through error rates ranging 10 to 50 percent:

```
%DO failrate = 10 %TO 50 %BY 10 ;

  < insert steps 4 and 5 here >

%END ;
```

ERROR FIELDS TO TEST THE PERFORMANCE OF THE SAMPLE

So ,what we have just done, is reconstruct the amount of our sampling units, that might be attributable to error, included a simulated range of variation. We then used a random selection tool to model different frequencies of error that an audit team might encounter in the field. In doing this we have not created one dataset to emulate the types of expected error, but several. In the previous example we created 5 datasets ranging from a rate of 10 percent to 50 percent of our sample having fraud, waste or abuse – generically referred to as "error." Because our random selections were taken from the previously modeled ranges of error amount we have successfully merged the impact of these two, very different forces that affect sample variance. We don't yet know the true sampling distribution we may encounter, but we have a test bed of data to work with in verifying the performance of any sample designs we might develop.

We refer to this test bed of data as our "Error Field." The advantage of having these error fields is that not only do they mimic the challenges a sample might encounter in the field but they contain a known quantity of dollar error that can be compared with the estimate gained by the sample.

DESIGNING THE STRUCTURE OF THE SAMPLE

The next step is to design a sampling structure that will control for factors that could reduce the predictability of variance or impart bias. We do this through selection design, stratification and selection controls. Instead of designing for one number set, we are designing for a range of number sets, as established in our error field. Given that audit samples often have a sampling distribution that is not quite Gaussian, we use replicated sampling to verify the performance of the sample design across a range of conditions. In this manner we have demonstrable proof that the theory behind applying known multiples of a standard error, adequately reflects what happens in nature. In cases where it doesn't, we can detect that and make adjustments.

Given that the sample results may well have ramifications on fiscal reporting, budgets or legal settlements in court, it is also helpful to have tangible evidence of how the sample behaves. While it takes more initial investment to adopt this technique, you will find that, over time, you acquire a much deeper sense of how samples perform, the things that drive design, and the techniques you need employ to contain skewness or unexpected tail conditions. With these tools in hand you will find that your sample design becomes an iterative process: modifying, tightening, loosening and retesting designs.

SELECTION DESIGN

Audit samples have to be fairly robust, and sometimes need to stand up under changes and adjustments. They sometimes serve teams of people with competing interests. If attorneys are involved they are rarely concerned about the niceties of things like selection probabilities so it is important to connect with them ahead of time and anticipate the various directions they might decide to take things. It is not unheard of to change horses in mid-stream and add new strata or blend two sample designs. Sample counts may be increased for non-statistical reasons. The original universe may turn out to include some records that have no bearing on the original question.

For this reason and others it is often good to stick with selection methods that involve equal probability of selection (EPSEM) within any one strata. Often this means sticking to a "proportional" sampling method where the count in each stratum that is roughly proportional to that stratum's fraction of the whole. The SAS code illustrated in the next section will illustrate how proportional sampling in SAS can be set up in a way that easily adapts to changes in sample count and design.

We have successfully applied probability proportionate to size methods (PPS) to audits that resulted in high-dollar settlements with financial institutions, but these tended to have limited, close-knit audit and legal teams. Response rates are usually very good in audit sampling so it's possible to make probability proportionate to size (PPS) samples work. It just runs the risk of an unnecessarily widening the margin of error if substitutions have to be (conservatively) selected and they preclude any back-tracking or additions and they preclude projecting percentages or counts. The SURVEYSELECT procedure in SAS allows for more than one approach to PPS sampling and these methods can be very powerful. They can also be very fragile in the face of changing goals or unexpected things in the data.

In certain cases one can use optimized samples such as Neyman samples in audit sampling, but they take special preparation and care. Optimized sampling uses a selection method that concentrates samples in the strata that will be most influential in the final design. Optimized sampling can be very effective in audit sampling and we have used them successfully, but the means of implementing them for audits is beyond the scope of this article.

So, in many cases the best selection method is a proportional random selection among strata. The primary control of variance is left to the stratification of the sample design. As is covered in the performance testing section, the method described herein employs replicated sampling and this can provide a powerful tool in explaining the performance of your chosen sample design.

STRATIFICATION

Early in setting up the error field you will set-up a skeletal foundation for your stratification design. As William Cochran said in his classic reference on sampling (Cochran, 1977):

> *The ideal variate for stratification is the value of y itself – the quantity to be measured in the survey. If we could stratify by the values of y, there would be no overlap between strata, and the variance within strata would be much smaller than the overall variance.*
>
> -- William G. Cochran

This is particularly true in audit sampling and in our case "y" is the original dollar value of the sampling item. We don't know the final error amount that will be found in the field but the original dollar value of the item provides the best way of ranking and grouping sampling items. Audit sampling occurs under a lot of limitations, but one of its distinct advantages is that it usually involves a finite, describable universe and we usually know the base value of each sampling item at the time we're doing the sample design.

Once you adopt replicated sampling to proof your sample designs and test the applicability of theory it becomes hard to resist an iterative design process where you test, tune and retest your stratification designs. Effective stratification often involves fine tuning the relative boundaries between strata, and it's can be helpful to rank the data by "y" and have SAS compute the exact breakpoint based on the quintile or percentile you specify. Below is some code that allows you to dictate relative rank were boundaries between strata occur and have SAS compute the literal dollar amounts involved. In this manner you can think theoretically while the machine does the practical work of implementing the breakpoints.

```
proc sort data= UNIVERSE   ; by amount ; run ;
proc univariate data= UNIVERSE  noprint   ;
   var amount ;
   output out=STRATA_BREAK_PTS pctlpts  =  25 50 75  pctlpre=a  ;
run ;
```

```
data STRATA_BREAK_PTS2 ; set STRATA_BREAK_PTS ;
   a20 = round(a25,1) ;  a50 = round(a50,1) ; a75 = round(a75,1) ;

   put "  if amount gt 0  then size_tier = '0-25pct' ; " ;
   put "  if amount ge "  a25 @40 "then size_tier = '25-50pct' ; " ;
   put "  if amount ge "  a50 @40 "then size_tier = '50-75pct' ; " ;
   put "  if amount ge "  a75 @40 "then size_tier = '75-100pct' ; " ;
   put  ;
run ;


data MODLPOP(drop=size_tier) ; set UNIVERSE end=last ;
  format strata size_tier $10. ;

  if amount gt 0        then size_tier = '0-25pct' ;
  if amount ge 42195    then size_tier = '25-50pct' ;
  if amount ge 130457   then size_tier = '50-75pct' ;
  if amount ge 5254605  then size_tier = '75-100pct' ;

  < other code >
run ;   %END ;
```

You'll notice that in the example above, we print the stratification code to the SAS log and the statistician still has to manually copy and paste them into the code. You could, of course, shoot the code into a text file and have the program automatically draw it back in as executable code. This is a good place, however, to require conscious, human intervention each time a design change is implemented. Please note that the dollar amounts and percentile points shown above are arbitrary breakpoints offered only as an example of how to structure the code. The actual breakpoints would be different, as the points shown above would likely *not* make a good sample design.

With respect to timing, we should also note, that this stratification process is typically inserted in the middle of defining the error field: after defining the dollar amount and before simulating the probability of error. Conceptually, stratification is modified in response to performance testing (below) until the best design is found. The entire process is rerun with each modification of the design, and for that reason we show the process here as a prelude to performance testing.

SAMPLE SIZE

Because we recommend the final design involve replicated sampling, we don't select a sample size until we have tested the performance of the sample with the error field and reproduced the true sampling distribution of the sample. We will use classic sample design formulas, however, to estimate an SRS sample size and to provide a point of comparison, but the final size is driven by what testing shows will conform to the stated confidence interval and adequately control larger values that might occur in the sampling distribution's tails.

PERFORMANCE TESTING: MODELING THE SAMPLING DISTRIBUTION

As was noted before, the interaction of various error rates, null values, selection probabilities, etc. involves an essentially infinite number of possibilities. While the probability distribution of a given sample design may be incalculable mathematically, we can quantify the combined effect of these factors by constructing the sampling distribution through replicated sampling. While it takes extra work in the beginning to set up processes which are essentially thousands of cycles of bootstrapping, the solid information about how the sample performs can greatly improve (and change) your approach to sampling.

TESTING A SERIES OF DESIGNS WITH REPLICATED SAMPLING

With the replicating function in SAS's sampling routines, the macro programming ability and the ability to engineer code that writes itself, SAS provides a very good platform for bootstrapping and replicated sampling. Once these systems have been set-up you can benefit from the initial investment of time and capital.

Below is a simplified implementation of replicated sampling or "boot-strapping" in SAS to establish various sampling distributions:

```
%let trials = 1000 ;
%let failrate = 30 ;
%let sampsize = 80 ;

proc sort data= MODLPOP2 ; by STRATA_var rec_id ; run ;

/*** (1) GENERATE A SAMPLE COUNT DESIGN FOR THE SAMPLE SIZE BEING
         TESTED. ****/
proc surveyselect noprint
  data= MODLPOP2
  method=srs
  seed = 1492
  n = &sampsize
  out = Sampl eSizes ;

 strata STRATA_var  /
    alloc=prop nosample
    allocmin = 2  /***set a minimum sample size of 2 per stratum.  ****/ ;
run ;

/**** (2) APPLY THE SAMPLE DESIGN CREATED ABOVE TO THE CHOSEN
          ERROR FIELD AND GENERATE A SERIES OF AUDIT FINDINGS
          IN ORDER TO ESTABLISH THE SAMPLING DISTRIBUTION. ****/

proc sort data= POP_err&failrate. ; by STRATA_var rec_id ; run ;
proc surveyselect noprint
  data= POP_err&failrate.
  method=srs
  seed = 1776
  n = SampleSizes
  rep = &trials
  out = TEST_SAMPLE&failrate. (rename=(replicate=sample)) ;
  strata STRATA_var  ;
run ;

/**** (3) COMPUTE THE PROJECTED RESULTS OBTAINED FROM
          THE REPLICATED SAMPLES.  **/

/**** Store the strata sizes to support the calculation of an FPC ***/
data STRATA_SIZES (rename=(total=_Total_)) ;
  set SampleSizes(keep=strata total)  ;
  do sample =1 to &trials. by 1 ; output ; end ;
run ;
proc surveymeans data=TEST_SAMPLE&failrate.  mean var nobs
                 total=STRATA_SIZES  ;
by sample ;
  var aud_diff ;
```

```
  strata strata ;
  weight samplingweight ;
ods output statistics = rep_STATs ;
run ;
```

Adding a REP = parameter to the second SURVEYSELECT procedure allows us to define a series of repeating sample selections to establish our bootstrap process. In practice you will find yourself doing dozens, even hundreds of replication groups and each tends to inform the other. For this reason, we can usually gain a good understanding of how the designs perform with 1,000 replications of each iteration. This is fortunate, because, the machine time required for 100 or more iterations during the design process can consume a meaningful amount of time. There is nothing wrong of course with using more iterations to test certain conditions.

The simple SURVEYSELECT code shown previously, can be automated by wrapping it in macro code similar to the following:

```
%DO sampsize = 70 %TO 120 %BY 5 ;
   %DO failrate = 10 %TO 50 %BY 10 ;

  < insert steps 1,2 and 3 of the replicating code here >

   %END ;
%END ;
```

EVALUATING THE PERFORMANCE OF YOUR DESIGNS

Once you have let your computer run several (hundred) thousand iterations using replicated sampling, you can evaluate the performance of the sample designs as follows:

1. Apply a margin of error by computing the upper and lower confidence intervals.
2. Review whether the lower/upper confidence intervals exceed/undershoot the known, true error amount.
3. Rank the results, and count the number that actually exceed/undershoot the true amounts
4. Compare the percent whose LCL/UCL exceeds/undershoots the true amount and compare it with the stated confidence interval.

Evaluating Conformity the Confidence Interval

One of the advantages of sample designs for audits and forensic estimations of damages is that we only care about the left tail of the probability distribution. Our primary goal is protecting the company being audited from a risk of over-estimating which exceeds the stated confidence interval. Our goal is to be able to say “We can say – with a one-sided confidence interval of 95 percent – that the amount of damages/fraud/waste/etc. is *at least* $XYZ dollar amount.”

In many cases the true sampling distribution will be lognormal. This means that extending out from the mean by some multiple of the std. error will not capture the same amount of the pdf on the left and right sides. With good sample design guided by replicated sampling we can return the sample distribution to something that approaches the normal distribution but, in general, audit samples tend to produce a slightly skewed sampling distribution. From a legal or auditing perspective we can live with that as long as the chance of overstating the projected amount remains less than the stated confidence interval.

One of the things you will learn when you start confirming the sampling distribution of your audit sample designs is that many times the risk of over-estimating meets or exceeds your confidence interval (which is good for the company being audited) but that the risk of under estimating can have a greater risk than you'd expect from your z-score and stated confidence interval. With the experience gained from modeling the sampling distributions of your designs, your skill at minimizing the risk of underestimating will improve. Audit samples, however, always have a tendency toward the lognormal.

You will also learn that on some rare occasions, certain error amounts combined with error rates approaching 50 percent do not quite conform to stated LCL using the traditionally calculated t-score to extend the standard error, regardless of the sample design. You may get an extra half a percent or so in the upper tail. In this case, classic sampling theory is insufficient and you will need to very slightly increase the t-score (and margin of error). This occurs with or without the use of replicated sampling to design your sample and it is only by reproducing the sampling distribution in this manner that you become aware of it.

Evaluating "Worst Case" Projections

While it is not yet a practice in the field we also evaluate how extreme projections can become in the outer tail of the sampling distribution. We do this because many times audit sampling results in a settlement involving money, or dollars returned to the treasury or other real-life consequences. Since sampling distributions can have subtle but significant variations in the tail of the pdf we also verify that the increase from a rare-on-in-a thousand event is less than 50 percent. Many times it is substantially less. While it is not standard practice in the field to do this we imagine that someday auditees may raise questions about how a sample design handles those rare extreme tail events. Hence, we add this as a second way of evaluating audit samples. Experience has shown that managing rare outer tail conditions can be at least as demanding on the design as traditional evaluation of the confidence interval.

CONCLUSION

With the advent of computerized methods and the increasing use of statistical samples in court, both the means and the need are growing with respect to verifying how a sample design performs in nature. Theory still provides vital insight into what drives sample design, but in audit settings it is preferable to be also be able to tangibly demonstrate how a sample design will perform.

Similar to the way handmade furniture gave way to furniture made by machines, this means that we must sometimes relinquish control to processes that may strike us as nothing more than the brute force of the machine. In 1945, however, renowned statistician R.A. Fisher looked upon the early exercises in replicated sampling and said the following (Statistics Dept. University of California, Davis):

> The use of the method of random sampling is theoretically sound. I may mention that its practicability, convenience and economy was demonstrated by an extensive series of crop-cutting experiments on paddy carried out by Hubback… they influenced greatly the development of my methods at Rothamsted
>
> –R.A. Fisher

Obviously the approach outlined herein requires an investment of time, infrastructure and research – though the resulting offer commensurate rewards. The sample code and methods and methods outlined herein are intended as an introduction. While, hopefully, they serve to inform, they are a mere sketch of the full effort required to make use of this approach.

REFERENCES

Cochran, W. G. (1977). *Sampling Techniques.* New York: John Wiley & Sons, Inc.

Kish, L. (1965). *Survey Sampling.* New York: John Wiley & Sons, Inc.

Lohr, S. L. (1999). *Sampling: Design and Analysis.* Pacific Grove, CA: Duxbury Press.

Statistics Dept. University of California, Davis. (n.d.). *Methodology and Theory for the Bootsrap.* Davis, California: UC, Davis.

Vickers, A. (2010). *What is a P-Value Anyway?* Boston, MA: Addison Wesley.

Wayne W. Daniel, J. C. (1983). *Business Statistics.* Houghton Mifflin, Company.

ACKNOWLEDGMENTS

A special thanks is extended to Clifton C. Cole, Director of the Integrated Data Analytics Division at HUD – OIG for sensing that something was missing in the traditional approach to audit sampling and for having the vision to step back and let his Statisticians develop a better way to do it.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Turner Bond, Statistician
HUD – Office of Inspector General
(202)407-3533 or (573)268-8385
tbond@hudoig.gov