Implementing A Nonlinear Optimization Procedure to Estimate Disclosure Risk

Betsy S. Greenberg
Management Science and Information Systems Department
University of Texas at Austin

Task

Disclosure risk is usually assessed by estimating the number of individuals in a population with unique characteristics. The problem is difficult because there are situations for which samples from two or more very different populations can be nearly identical. This project will examine an optimization procedure to estimate the number of unique individuals in a population based on sample information. The resulting optimization program has multiple solutions corresponding to different populations that could have been the source of the sample data. Many if not all local solutions can be found using a new global optimization algorithm called OptQuest NLP (OQNLP). The goal of this project was to modify the program to work on large practical problems, test the procedure with public use Census data, and describe the result and its limitations.

Background

Sets of characteristic values that are unique in a sample or data set are called sample uniques and sets that are unique in a population are called population uniques. Sample uniques may or may not be population uniques. All methods proposed for measuring disclosure risk involve estimating the proportion of the sample uniques that correspond to population uniques or the number of population uniques.

Various models have been proposed in the literature for estimating the number of population uniques from a sample of data. The Poisson-Gamma (Bethlehem, et.al., 1990), Poisson-lognormal (Skinner and Holmes, 1993), Dirichlet-multinomial (Tekemura, 1999), and

the negative binomial (Chen and Keller-McNulty, 1998) models have all been proposed. These models assume that class sizes in a population follow a particular distribution. The performance of these models depends on how well the population follows the model. Unfortunately, performance is often poor, especially for low sampling fractions.

Greenberg and Zayatz (1992) proposed a procedure that is not dependent on a model for the population of class sizes. Instead, they use the class sizes in the sample as an estimate for the class sizes in the population and use the result to estimate the probability that a unique in the sample is a unique in the population. The methods considered here are extensions of G&Z.

Let μ_i be the number of classes of size *i* in the population and let p_i be the proportion of classes that are of size i in the population. That is,

$$p_i = \mu_i \Big/ \sum_{all\ i} \mu_i \,. \tag{1}$$

Let m_i be the number of classes of size *j* in the sample. If the p_i's are known, we can calculate

$$P(i_p \mid j_s) = \frac{p_i P(j_s \mid i_p)}{\sum_{\text{all } i \geq j} p_i P(j_s \mid i_p)}, \text{ and} \tag{2}$$

$$\mu_i = \frac{\sum_{j \leq i} m_j P(i_p \mid j_s)}{1 - P(0_s \mid i_p)}, \tag{3}$$

where $P(j_s \mid i_p)$ is a hypergeometric probability.

$$P(j_s \mid i_p) = \frac{\binom{i}{j}\binom{N_p - i}{N_s - j}}{\binom{N_p}{N_s}}, \tag{4}$$

where N_p is the size of the population and N_s is the size of the sample.

$P(i_p \mid j_s)$ is the probability that a class of size *j* appearing in the sample came from a

class of size *i* in the population and can be calculated using Bayes' Rule in (2). Equation (3) is a method of moments estimate for μ_i. Greenberg and Zayatz (1992) estimate p_i from the observed class proportions seen in the sample, so $p_i = m_i \Big/ \sum_{all\ i} m_i$. These estimates are then used to solve for $P(i_p \mid j_s)$ and μ_1, the number of uniques in the population. The algorithm can potentially be improved by solving for all of the μ_i in (3) and using the revised estimates to solve for p_i $P(i_p \mid j_s)$ and μ_i again. We call this the 1-step recursive procedure. The process can also be repeated until the procedure converges as described in Greenberg (2002). The upper limit on the sums in (1), (2), and (3) is selected so that the resulting class sizes are consistent with the population size. That is,

$$N_p = \sum_{i=1}^{M} i\mu_i, \tag{5}$$

is satisfied as closely as possible. The resulting values of $\mu_1, \mu_2, \cdots$ provide estimates for the number of uniques in the population, the number of pairs in the population, etc. Since data coming from rarely occurring sets also has potential for disclosure, this is more useful than only estimating the number of uniques.

The recursive algorithm, considered in Greenberg (2002) will converge to a population from which the sample data could have been obtained. The estimates provided when sampling rates are high to moderate are superior to those provided by other methods. Unfortunately, for low sampling rates, there are situations for which samples from two or more very different populations can be nearly identical. For such examples, the recursive algorithm may converge to the "wrong" population. This problem is illustrated in Greenberg (2002) for two populations, one with no uniques and another with many uniques.

In situations where two very different populations could have generated the same sample, it will be very difficult, if not impossible, to find a method guaranteed to work well unless additional information is used. When prior information is available, a starting point different from the one based on the sample could be used, resulting in a much better solution. The recursive algorithm converges to different solutions depending on the starting point. In the absence of reliable prior information, it should be helpful to arbitrarily choose a variety of starting points to see whether alternate solutions exist. In practice, although we can't determine which is the correct solution, it is useful in assessing disclosure risk to know that the data *may* have come from two or more different populations.

Formulation of the Optimization Model

To eliminate p_i from our formulation, we substitute (1) into (2), yielding

$$P(i_p \mid j_s) = \mu_i P(j_s \mid i_p) / \sum_{k \geq j} \mu_k P(j_s \mid k_p), \text{ for all } (i,j) \text{ with } i \leq M,\ j \leq S,\ j \leq i \qquad (6)$$

where S is the size of the largest class in the sample and M is the size of the largest class in the population. The relaxed estimation equations (3) are:

$$\mu_i[1 - P(0_s \mid i_p)] - \sum_{j=1}^{i} m_j P(i_p \mid j_s) = pdev_i - ndev_i,\ i = 1,...,M \qquad (7)$$

where $pdev_i$ and $ndev_i$ are nonnegative "deviation" variables representing the positive and negative errors in the ith equation.

To reflect the fact that classes of size j in the sample, must arise from classes of size j or larger in the population, we impose the constraints

$$\sum_{i \geq j} \mu_i \geq m_j,\ j = 1,...,S \qquad (8)$$

In addition, all variables μ_i, $P(i_p \mid j_s)$, $pdev_i$, $ndev_i$ must be nonnegative. The objective is to minimize some norm of the residuals in equation (7). If the L_1 norm, the sum of absolute residuals is used, the objective is:

$$\text{minimize} \qquad abserr = \sum_{i=1}^{M} (pdev_i + ndev_i) \tag{9}$$

Alternatively, if the L_2 norm is used, the objective would be to minimize the sum of the squared deviations. If the $L_{\max}$ norm is used, the objective would be minimize the largest deviation. For this project, the L_1 norm was used.

The resulting model minimizing (9) subject to (5), (6), (7), and (8) is an overdetermined set of nonlinear equations. The optimization program attempts to find an approximate solution to the set of equations by minimizing the sum of absolute residuals. Since the resulting nonlinear program is nonconvex, it will have several local minima. Many solutions can be found using a new global optimization algorithm called OptQuest NLP (OQNLP). Initial testing with artificial data, described in Greenberg and Lasdon (2003), showed that the model does generate multiple solutions. In some cases, especially when sampling rates are high, only one or two populations are identified. When sampling rates are lower, more potential populations are identified, often with very different values for the number of population uniques. The solutions generated in all cases have the property that the expected samples are nearly equal to the input sample.

There is not likely to be a method that could accurately determine from which population the sample was actually selected. However, when all potential solutions are identified, rather than just one that may be incorrect, disclosure risk can be more accurately assessed. Similarly, when multiple solutions are found for the number of species in a population, it will provide information about the degree of uncertainty in the estimates.

Software Description

The optimization model was coded using The General Algebraic Modeling System (GAMS) and solved using OQNLP with CONOPT. GAMS is specifically designed for modeling linear, nonlinear and mixed integer optimization problems. Additional information about GAMS is available at http://www.gams.com. Instances of the model are provided at http://www.gamsworld.org/global/globallib/globalstat.htm under the names bayes2_10, bayes2_20, bayes2_30, and bayes2_50. The full model is provided in Appendix A.

OQNLP is a multi-start heuristic algorithm designed to find global optima of smooth constrained nonlinear programs. Multi-start means that OQNLP calls an NLP solver from multiple starting points, and keeps track of all feasible solutions found by the solver. The starting points are computed by a scatter search implementation called OptQuest. The scatter search algorithm operates on a set of points, called reference points, which constitute good solutions from previous solution efforts. The approach systematically generates new combinations of the reference points to create new points, each of which may be mapped into an associated feasible point. The process of generating new points is intended to balance two important, but often contradictory requirements: intensification and diversification of the new points. Additional information about OQNLP is available at http://www.gams.com/dd/docs/solvers/oqnlp.pdf.

The output for this project was produced with default parameter values for OQNLP options and tolerances, except the maximum runtime was increased (from 1000 to 20000 seconds) and the distance and merit filters were turned off to obtain as many local solutions as possible. The OQNLP options file has the default value of 1000 solver calls, so it was possible for that many solutions to be generated.

The NLP solved called by OQNLP was CONOPT. CONOPT is a solver for large-scale nonlinear optimization (NLP) developed and maintained by ARKI Consulting & Development A/S in Bagsvaerd, Denmark. It has been under continuous development for over 25 years. CONOPT is a feasible path solver based on the GRG method with many newer extensions. CONOPT has been designed to be efficient and reliable for a broad class of models, especially for models where feasibility is difficult to achieve. Additional information about CONOPT is available at http://www.gams.com/docs/conopt3.pdf.

Test Populations

Two populations were considered for this project. The first population is described in Table 2 of "Estimation of the percent of unique population elements on a microdata file using the sample" by L.V. Zayatz (Census/SRD/RR-91/08). The population had 56,372 individuals and 22,026 of these are in unique categories. Class sizes in the population ranged up to 298. According to Steel (2003), a population with so many individuals in unique categories would be unpublishable as a table for a county or tract publication. Table 3 of Census/SRD/RR-91/08 provides one sample from this population. Nineteen additional samples were randomly generated for testing.

The second population considered was a similar population with 56,376 individuals. Steel (2003) coarsened the categories used to tabulate the population data so that the disclosure risk would be lower. Only 1,175 individuals in the second population were in unique categories. Class size in the second population ranged up to 1835. Twenty samples were randomly selected from this population for testing.

A sampling rate of 1 out of 6 was used to generate all populations including the one in Table 3 of Census/SRD/RR-91/08. The data for each sample is included in the GAMS model in

Appendix A. The following line from the program corresponds to the sample described in Table 3 of Census/SRD/RR-91/08.

```
/   1    5563,     2     591,     3     171,     4   97,     5   54,     6   44,     7   29,
8   23,     9   10,    10   10,    11   10,    12   12,    13    5,    14    5,    15    3,    16
1,    17    3,    18    1,    19    1,    22    1,    66    1/;
```

That is, there were 5,563 uniques, 591 pairs, 171 groups of three, … and one group of 66 in the sample. Lines starting with an asterisks (*) in the GAMS code are comments, so samples can be selected by moving the asterisks.

Modifications needed for census problems

Greenberg and Lasdon (2003) considered examples with population class sizes of no more than 22. This project considered the populations with much larger class sizes (298 and 1,835). Solving problems of this magnitude required significant modifications to the programs.

1. Model reduction

The optimization model minimizing (9) subject to (5), (6), (7), and (8) has $S(2M-S+1)/2+3M$ variables with non-negativity constraints and $S(2M-S+1)/2+S+M+1$ additional constraints. Eliminating $P(i_p \mid j_s)$ can reduce the model size. To do this, define Em_j, the expected number of classes of size j in the sample:

$$\mathrm{E}m_j = \sum_{i=j}^{M} \mu_i P(j_s \mid i_p) \qquad j = 1,...,S \tag{10}$$

Substitute (10) into (6) to obtain

$$P(i_p \mid j_s) = \mu_i P(j_s \mid i_p) / \mathrm{E}m_j, \qquad \text{for all } (i, j) \text{ with } i \le M,\ j \le S,\ j \le i$$

which we substitute into (7) to eliminate $P(i_p \mid j_s)$. We obtain

$$\mu_i[1 - P(0_s \mid i_p)] - \mu_i \sum_{j=1}^{i} m_j \frac{P(j_s \mid i_p)}{\mathrm{E}m_j} = pdev_i - ndev_i, \qquad i = 1,...,M$$

which we simplify by noting that $\sum_{j=0}^{i} P\left(j_s \mid i_p\right) = 1$. Finally, we obtain

$$\mu_i \sum_{j=1}^{i} P\left(j_s \mid i_p\right)\left(1 - \frac{m_j}{\mathrm{E}m_j}\right) = pdev_i - ndev_i, \qquad i = 1,...,M \tag{11}$$

It is necessary to impose small positive lower bounds on $\mathrm{E}m_j$ because it appears in the denominator of (11):

$$\mathrm{E}m_j \geq eps \qquad\qquad j = 1,..., S \tag{12}$$

where *eps* = 1.E-10 has sufficed for this project.

The formulation minimizing (9), subject to (5), (8), (10), (11) and (12) has $3M + S$ variables with non-negativity constraints and $3S + M + 1$ additional constraints. For the first population with class sizes up to 298 in the population, this reduction reduces the size of the problem from one with more than 18,000 variables and 14,000 constraints to one with less than 1000 variables and 500 constraints. For the second populations with class sizes up to 1,835 the improvement if even more substantial. The problem size is reduced from a model with over half a million variables and constraints to one with fewer than 3000 constraints and 6000 variables.

2. Smoothing Constraint

Optimal solutions of the above NLP estimation problem have many components of the errors (where $error_i = pdev_i - ndev_i$) in the estimator equations (11) equal to zero. This causes many components of the estimate vector $\hat{\mu}_1, \cdots, \hat{\mu}_M$ to be zero. Examination of (11) shows that the error in the *i*th estimator equation is the product of μ_i and the weighted sum of deviations between the expected sample for the population and the actual sample. Hence if the *i*th error is nonzero, μ_i must be nonzero. If the *i*th error is zero, μ_i can be nonzero if the second factor in

the product is zero. However it is rare for the second term to vanish, so zero errors usually imply zero values for μ_i.

This "choppiness" leads to poor estimates when the true class size distribution is smooth. This can be remedied by imposing smoothness constraints on the μ_i. Steel (1999) suggested that a monotonicity constraint

$$\mu_i \geq \mu_{i+1} \qquad i = 1,...,M\text{-}1 \tag{13}$$

would be appropriate. (13) was used to correct for choppiness and to reflect the belief that the populations with disclosure risk have a high number of uniques, fewer pairs, etc. This adds an additional M-1 constraints to the optimization problem.

3. Reducing solver problems leading to infeasibility

Numerical problems occur because the model involves both large numbers, such as the population size and small probabilities. As a result, CONOPT initially failed to find a feasible solution for one of the samples generated from the first population and all of the samples initially tested for the second population. To improve the solver performance, the following changes were made.

Values of $P(j_s \mid i_p)$ less than a small threshold were set to zero. This reduced the number of non-zero model coefficients, making the problem easier to solve. Sometimes CONOPT fails when a derivative is very large. CONOPT uses a default value of RTMAXJ=1.0E5. If derivatives larger than this default value are encountered, CONOPT fails to find a solution. By increasing the value of RTMAXJ to 1.0E-8 (in the CONOPT options file) and setting THRESH=1.E-6 (in the GAMS model provided in Appendix B), all samples for both populations generated solutions.

For the second population considered in this project, the class size in the population ranged up to 1,835, resulting in classes ranging up to 346 in one of the samples. To reduce the size of the problem, a few problems were tested by allowing class sizes in the sample to be 50 or less. The size of the sample falling into classes of size 50 or smaller was calculated and then the population was assumed to be equal to six times that amount. Similarly, the size of the largest class in the population was set to be equal to 350. The maximum values of 50 and 350 were set somewhat arbitrarily, however initial test results showed that the results from the reduced problem were very similar to the results from the full problem.

Results

For all 20 cases from each population, results were obtained from the optimization model. A LOCALS.OUT file was generated with the objective and variable values for each feasible solution. When multiple solutions are generated, it is not clear how to use the results to estimate disclosure risk. One possibility is to be conservative and use the solution with the largest value for μ_1, the number of uniques. Other possibilities are to use the mean or median of the solutions obtained. The mean, median, and maximum were calculated using all of the solutions reported that had objective values no larger than 10 times as large as the best objective value. The cutoff of 10 was selected arbitrarily. The solution with the best objective value, the mean, median, and maximum in addition to the number of solution used for these calculations is listed in Tables 1 and 2. As expected, the solution with the lowest objective value is usually not the one that is closest to the correct value.

Although duplicate solutions were deleted from the LOCALS file, many of the solutions were very similar. As a result, the number of different values for the estimate of μ_1 is much lower than the number of solutions. The number of different solutions (for the number of uniques) is also listed in Tables 1 and 2.

The results for the optimization model appear to be too variable to use. The estimates are especially poor for the second population described in Appendix B. Comparing the results for the two populations, the objective values for the first population were much lower, averaging about 1.6 for the first population and about 53 for the second. Similarly, many more solutions were obtained for the first population, an average of 511 for the first population, compared to only 38 for the second. Therefore, although the second population has fewer uniques, it appears to be more difficult to estimate that number using the optimization method considered in this project.

In both cases, the objective values obtained by GAMS/OQNLP/CONOPT are considerably lower than the values obtained by calculating Em_j using the true population values in (10) and then calculating the deviation values using sample values in (11) and adding them up as in (9). On average, these values were about 55 for the first population and 98 for the second.

For comparison, the estimates obtained using the method of Greenberg and Zayatz (G&Z), 1-step of the recursive procedure, and two other methods (Bethlehem, et.al. and Hashino) are given in Tables 1 and 2. Both Bethlehem, et.al. and Hashino appear to be very biased. G&Z and the solutions from the 1-step recursive procedure appear to be better. For the first population, the 1-step procedure improves the result from the G&Z initial solution in all 20 samples. Both the G&Z and the 1-step estimates are consistent overestimates for population 1. This is a desirable property, since it would be preferable to overestimate disclosure risk. The average value of the bias is 2564 for G&Z and 1689 for the 1-step procedure.

For the second population, the 1-step procedure improves the result in the majority (15 out of 20) of the samples. That is, the results from the 1-step procedure are usually closer to the true value. G&Z estimates for this population are biased high in 19 out of 20 cases for an

average bias of 123. The estimates from the 1-step procedure are biased high in 18 out of the 20 cases for an average bias of 105.

Recommendations

The variability in the results from the optimization procedure may be due to the difficulties in solving the nonlinear program or it may be due to the lack of information in the sample. Further study is needed to see if the optimization procedure can be improved. If solutions can be obtained with lower objective values, it is likely that the results would be more useful.

Further study is needed to see if the 1-step recursive procedure is useful and whether allowing the procedure to converge would further improve the solutions.

References

Bethlehem, J.G., Keller, W.J., and Pannekoek, J. (1990). Disclosure Control for Microdata. *Journal of the American Statistical Association,* 85, 38-45.

Chen, G. and Keller-McNulty, S. (1998). Estimation of Identification Disclosure Risk in Microdata. *Journal of Official Statistics*, 14, 79-95.

Greenberg, B.G. and Zayatz, L.V. (1992). Measuring Risk in Public Use Microdata Files. *Statistica Neerlandica*, 46, 33-48.

Greenberg, B.S. (2002). A Recursive Algorithm to Estimate Disclosure Risk. Submitted for possible publication to *Journal of Official Statistics*.

Greenberg, B.S. and Lasdon, L.S. (2003). A Global Optimization Procedure to Estimate Class Sizes. Submitted for possible publication to *The Journal of Global Optimization*.

Hoshino, N. (2001), Applying Pittman's Sampling Formula to Microdata Disclosure Risk Assessment, *Journal of Official Statistics*, 17, 499-520.

Skinner, C.J. and Holmes, D.J. (1993). Modelling Population Uniqueness. Proceedings of the

International Seminar on Statistical Confidentiality. Statistical Office of the European Communities, Luxembourg, 175-199.

Steel, Philip M. (1999), "A new estimation for the number of unique population elements based on the observed sample", *ASA Proceedings of the Section on Government Statistics and Section on Social Statistics*, 80-85

Steel, P. (2003). Personal communication.

Takemura (1999), Some superpopulation models for estimating the number of population uniques, *Statistical data protection - Proceedings of the conference, Lisbon, 25 to 27 March 1998 - 1999 edition*, Office for Official Publications of the European Communities, Luxembourg, 59-76.

Zayatz, L.V. (1991). Estimation of the Percent of Unique Population Elements in Microdata File Using the Sample. Statistical Research Division Report Series, Census/SRD/RR-91/08.

Table 1 – Results for First Population (correct number of uniques = 22026)

Sample	*Solution w/best objective*	*Mean*	*Median*	*Max*	*# Solutions*	*# Different Solutions*	*G&Z*	*1 step*	*Hoshino*	*Bethlehem*
1	15177	19600	20410	21656	47	47	24508	23606	29856	10918
2	18693	19868	18707	23988	652	9	24865	24013	29377	11663
3	23987	23426	23955	24019	297	15	25189	24386	29861	11679
4	23716	24020	23717	25163	629	2	25294	24496	29628	11748
5	25075	24003	24888	25205	547	29	24612	23689	29365	11242
6	25315	25109	25315	25315	601	12	24464	23533	29487	11546
7	23965	23895	23965	23965	606	2	24127	23224	29134	11670
8	22969	23735	24195	25338	178	149	24492	23598	29272	11115
9	23811	22338	22093	23947	631	28	24448	23629	28445	11274
10	18000	18002	18000	18008	635	2	24694	23923	29222	11253
11	24105	24105	24105	24105	614	1	24413	23492	29626	11604
12	26307	26283	26296	26307	631	12	24044	23186	28234	10817
13	25105	22033	22903	25105	452	88	24842	24016	29523	11424
14	15094	15094	15125	22462	552	13	24511	23635	29373	11415
15	21108	18450	17610	21108	627	11	24839	23924	29608	12041
16	27168	27168	27168	27168	615	1	24397	23457	30332	11605
17	21602	20836	20827	21602	657	5	23556	22562	28519	10925
18	20417	20146	20417	20417	622	4	25118	24291	29785	11569
19	17872	17872	17872	17872	623	1	24941	24074	29689	11765
20	25335	25335	25335	25335	1	1	24449	23567	29521	11023

Table 2 – Results for Second Population (correct number of uniques = 1175)

Sample	*Solution w/best objective*	*Mean*	*Median*	*Max*	*# Solutions*	*# Different Solutions*	*G&Z*	*1 step*	*Hoshino*	*Bethlehem*
1	519	519	519	519	31	1	1355	1344	2036	634
2	558	1109	565	2616	18	6	1352	1354	2062	609
3	565	565	565	565	29	1	1317	1317	1913	628
4	617	617	617	617	28	1	1262	1249	1895	615
5	1341	908	895	1341	35	2	1265	1254	1875	597
6	2928	2928	2928	2928	41	1	1458	1468	2032	639
7	752	867	613	2261	29	10	1199	1159	1942	609
8	2394	1285	1008	2394	30	3	1257	1237	1911	608
9	543	970	543	2588	31	6	1350	1360	1946	616
10	830	872	830	2443	42	3	1401	1389	2082	610
11	453	486	449	1158	114	7	1304	1283	1996	648
12	947	941	947	947	23	2	1313	1289	2013	625
13	522	522	522	522	27	1	1290	1274	1966	612
14	962	962	962	962	49	1	1323	1319	1944	625
15	1041	799	776	1355	36	3	1284	1265	1947	615
16	1292	698	645	1407	33	5	1238	1196	2000	608
17	420	463	420	1911	37	4	1227	1177	2042	622
18	551	810	554	2189	32	9	1166	1128	1873	623
19	473	476	473	492	51	2	1267	1232	1991	637
20	562	564	562	572	35	3	1330	1304	2052	640

Appendix B - GAMS Code

```
* This model estimates the class size distribution of a population using the
* same distribution in a random sample from that population. The estimator
* uses Bayesian principles
* Developed by Leon Lasdon and Betsy Greenberg
* Last update 8/25/03
* this version assumes the population class size distribution is monotone

Options limcol = 0, limrow = 0;

Sets i population index / 1*298/
     j sample index     / 1*66/
       alias (i,k);

Parameters
     m(j)  num size j classes in sample
* population1
* sample from Table 3
 /   1    5563,     2     591,     3     171,     4   97,     5   54,     6   44,
7   29,     8   23,     9   10,    10   10,    11   10,    12   12,    13    5,    14
5,    15    3,    16    1,    17    3,    18    1,    19    1,    22    1,    66
1/;
* sample 2
* /1     5621, 2     541, 3     208, 4     107, 5     58, 6     39, 7     17, 8
21, 9     14, 10     13, 11     13, 12     5, 13     6, 14     4, 15     2, 16     2,
17     2, 18     1, 20     2, 22     1, 48     1 /;
* sample 3
* /1     5657, 2     545, 3     180, 4     96, 5     65, 6     42, 7     27, 8     20,
9     14, 10     14, 11     6, 12     7, 13     8, 14     5, 15     2, 17     2, 18
1, 21     1, 22     1, 24     1, 47     1 /;
* sample 4
* /1     5674, 2     525, 3     207, 4     94, 5     60, 6     36, 7     26, 8     26,
9     13, 10     14, 11     9, 12     7, 13     3, 14     1, 16     3, 17     3, 18
3, 19     1, 21     1, 24     1, 46     1 /;
* sample 5
* /1     5598, 2     559, 3     201, 4     97, 5     70, 6     47, 7     31, 8     13,
9     15, 10     9, 11     10, 12     7, 13     1, 14     3, 15     1, 16     4, 17
2, 21     1, 25     1, 26     1, 64     1  /;
* sample 6
* /1     5573, 2     578, 3     208, 4     84, 5     64, 6     41, 7     33, 8     18,
9     13, 10     11, 11     6, 12     6, 13     3, 14     3, 15     5, 16     2, 17
2, 18     4, 21     1, 25     1, 48     1/;
* sample 7
* /1     5525, 2     578, 3     198, 4     95, 5     65, 6     46, 7     20, 8     29,
9     14, 10     13, 11     4, 12     8, 13     9, 14     3, 15     3, 16     3, 20
1, 21     1, 23     1, 43     1/;
* sample 8
* /1     5570, 2     559, 3     211, 4     94, 5     61, 6     33, 7     29, 8     19,
9     18, 10     11, 11     13, 12     8, 13     6, 14     3, 15     2, 17     1, 19
1, 20     1, 23     1, 25     1, 63     1/;
* samp9
```

```
* /1     5557, 2     537, 3     203, 4     103, 5     69, 6     36, 7     25, 8
25, 9     12, 10     16, 11     9,  12     6, 13     9, 14     1, 15     5, 16     2,
17     1, 21     1, 23     1, 57     1 /;
* samp10
* /1     5579, 2     535, 3     192, 4     102, 5     65, 6     35, 7     27, 8
25, 9     13, 10     14, 11     9, 12     5, 13     8, 14     3, 15     1, 16     3,
17     2, 18     2, 19     1, 20     2, 21     1, 51     1 /;
* samp11
* /1     5562, 2     589, 3     194, 4     85, 5     63, 6     46, 7     26, 8     21,
9     13, 10     9, 11     6, 12     6, 13     9, 14     8, 15     2, 16     2, 17
2, 21     1, 22     1, 23     1, 44     1  /;
* samp 12
* /1     5500, 2     552, 3     217, 4     98, 5     52, 6     40, 7     43, 8     26,
9     9, 10     12, 11     5, 12     4, 13     3, 14     4, 16     2, 17     4, 18
3, 20     1, 22     2, 23     1, 29     1, 58     1/;
* samp 13
* /1     5609, 2     548, 3     196, 4     96, 5     61, 6     47, 7     18, 8     20,
9     15, 10     12, 11     10,  12     6, 13     8, 14     3, 15     1, 16     5, 17
1, 18     1, 22     2, 28     1, 48     1/;
* samp 14
* /1     5570, 2     564, 3     194, 4     109, 5     65, 6     32, 7     23, 8
19, 9     17, 10     10, 11     11, 12     8, 13     5, 14     3, 15     1, 17     2,
18     1, 19     2, 20     1, 21     1, 22     1, 23     1, 24     1, 45     1  /;
* samp 15
* /1     5633, 2     562, 3     195, 4     106, 5     66, 6     35, 7     29, 8
19, 9     14, 10     12, 11     10, 12     5, 13     6, 14     3, 15     2,  16     1,
17     3, 20     1, 24     1, 45     1/;
* samp 16
* /1     5571, 2     552, 3     236, 4     107, 5     53, 6     41, 7     26,  8
18, 9     8, 10     8, 11     8, 12     7, 13     5, 14     3, 15     6, 16     3, 17
2, 18     1, 20     1, 22     1, 24     1, 48     1 /;
* samp 17
* /1     5452, 2     608, 3     189, 4     105, 5     72, 6     44, 7     28, 8
25, 9     11, 10     8, 11     2, 12     6, 13     8, 14     3, 15     1, 16     4, 18
1, 20     1, 21     2, 22     2, 26     1, 60     1/;
* samp 18
* /1     5653, 2     546, 3     178, 4     104, 5     60, 6     47, 7     27, 8
19, 9     17, 10     12, 11     10, 12     6, 13     3, 14     5, 15     2, 16     1,
17     2, 19     1, 22     1, 25     1, 55     1/;
* samp 19
* /1     5632, 2     556, 3     197, 4     104, 5     55, 6     30, 7     29, 8
16, 9     22, 10     12, 11     11, 12     8, 13     3, 14     4, 15     2, 16     3,
17     3, 18     1, 21     1, 48     1/;
* samp 20
* /1     5554, 2     575, 3     200, 4     84, 5     53, 6     42, 7     31, 8     20,
9     11, 10     17, 11     7, 12     7, 13     9, 14     5, 15     2, 16     5, 17
1, 18     1, 26     1, 61     1/;

* samp1 from population 2
* /1     632, 2     187, 3     89, 4     61, 5     42, 6     33, 7     33, 8     20, 9
12, 10     16, 11     17, 12     13, 13     16, 14     9, 15     8, 16     8, 17
12, 18     5, 19     2, 20     8, 21     3, 22     7, 23     5, 24     2, 25     5, 26
6, 27     3, 28     2, 29     2, 30     1, 33     1, 34     4, 35     2, 36     1, 37
1, 38     1, 40     2, 41     2, 43     2, 44     2, 45     1, 48     1, 49     2, 51
```

```
1, 52     1, 54     1, 59     1, 60     1, 62     2, 64     1, 66     1, 67     1, 72
1, 74     1, 76     2, 82     1, 85     1, 86     1, 87     2, 93     1, 94     1, 101
1, 104     1, 106     2, 114     1, 115     1, 123     1, 130     1, 132     1, 137
1, 138     1, 144     1, 179     1, 182     1, 312     1     /;
* samp2
* /1 625 ,   2 176 ,   3 79 ,   4 62 ,   5 53 ,   6 29 ,   7 37 ,   8 18 ,   9 13 , 10
17 ,   11 12 ,   12 8 ,   13 17 ,   14 9 ,   15 6 ,   16 5 ,   17 7 ,   18 5 , 19 15 ,
20 6 ,   21 7 ,   22 3 ,   23 3 ,   24 8 ,   25 4 ,   26 1 ,   27 2 , 28 2 ,   29 1 ,
30 2 ,   31 1 ,   32 5 ,   33 1 ,   34 3 ,   35 2 ,   36 1 , 37 1 ,   38 1 ,   40 2 ,
41 1 ,   43 1 ,   44 2 ,   45 1 ,   46 1 ,   47 1 , 48 1 ,   49 1,   52 1 ,   55 1 ,
57 1 ,   58 1 ,   59 2 ,   63 1 ,   64 1 , 65 1 ,   66 1 ,   70 1 ,   73 1 ,   78 1 ,
81 1 ,   82 1 ,   87 1 ,   89 1 , 92 1 ,   93 2 ,   103 1 ,   105 1 ,   113 1 ,   116
1 ,   118 1 ,   124 1 , 128 1 ,   134 1 ,   136 2 ,   138 1 ,   161 1 ,   179 1 ,
205 1 ,   321 1 /;
* samp3
* /1 617 ,   2 169 ,   3 88 ,   4 58 ,   5 48 ,   6 39 ,   7 39 ,   8 20 ,   9 12 , 10
13 ,   11 14 ,   12 14 ,   13 10 ,   14 6 ,   15 7 ,   16 10 ,   17 6 , 18 10 ,   19 2
,   20 5 ,   21 5 ,   22 9 ,   23 7 ,   24 4 ,   25 2 ,   26 2 , 27 3 ,   28 1 ,   29
2 ,   30 1 ,   32 6 ,   34 2 ,   35 4 ,   36 4 ,   39 3 , 44 1 ,   46 1 ,   47 2 ,
48 1 ,   49 1 ,   50 2 ,   52 1 ,   54 2 ,   55 3 , 58 1 ,   59 1 ,   61 1 ,   62 1 ,
66 1 ,   68 1 ,   81 1 ,   84 2 ,   86 2 , 87 2 ,   90 1 ,   91 1 ,   93 1 ,   102 1 ,
103 1 ,   105 2 ,   109 1 , 113 1 ,   118 1 ,   120 1 ,   137 1 ,   141 1 ,   146 1 ,
147 1 ,   181 1 , 190 1 ,   293 1 /;
* samp4
* /1 605 ,   2 171 ,   3 93 ,   4 69 ,   5 43 ,   6 33 ,   7 29 ,   8 27 ,   9 20 , 10
13 ,   11 15 ,   12 10 ,   13 9 ,   14 9 ,   15 9 ,   16 7 ,   17 11 ,   18 3 , 19 5 ,
20 6 ,   21 3 ,   22 3 ,   23 5 ,   24 3 ,   25 3 ,   26 7 ,   27 1 , 28 2 ,   29 4 ,
30 2 ,   31 3 ,   32 1 ,   33 5 ,   34 2 ,   36 3 ,   38 2 , 40 3 ,   41 1 ,   44 2 ,
45 1 ,   47 1 ,   53 1 ,   55 1 ,   56 1 ,   59 2 , 62 1 ,   63 1 ,   66 2 ,   69 1 ,
71 1 ,   73 1 ,   75 1 ,   80 2 ,   81 1 , 83 1 ,   86 1 ,   88 1 ,   99 1 ,   101 1 ,
102 1 ,   107 1 ,   109 1 ,   110 1 , 112 1 ,   120 1 ,   123 1 ,   125 1 ,   130 1 ,
132 1 ,   139 1 ,   142 1 , 151 1 ,   200 1 ,   211 1 ,   268 1 /;
* samp5
* /1 600 ,   2 166 ,   3 105 ,   4 57 ,   5 44 ,   6 31 ,   7 28 ,   8 23 ,   9 13 ,
10 17 ,   11 11 ,   12 12 ,   13 14 ,   14 12 ,   15 4 ,   16 9 ,   17 6 ,   18 6 , 19
7 ,   20 5 ,   21 6 ,   22 2 ,   23 7 ,   24 3 ,   25 6 ,   26 2 ,   28 4 , 29 5 ,
30 2 ,   31 1 ,   32 4 ,   34 1 ,   35 3 ,   37 1 ,   38 1 ,   39 2 , 40 2 ,   41 1 ,
43 2 ,   44 1 ,   46 2 ,   47 1 ,   48 1 ,   51 1 ,   57 2 , 59 1 ,   62 2 ,   63 2 ,
65 1 ,   67 1 ,   71 1 ,   75 1 ,   80 2 ,   82 2 , 83 2 ,   87 1 ,   89 1 ,   91 2 ,
92 1 ,   106 1 ,   107 1 ,   110 1 ,   112 1 , 114 1 ,   119 1 ,   130 1 ,   132 1 ,
140 1 ,   143 1 ,   145 1 ,   180 1 , 200 1 ,   345 1 /;
* samp6
* /1 659 ,   2 154 ,   3 102 ,   4 81 ,   5 50 ,   6 33 ,   7 17 ,   8 18 ,   9 20 ,
10 15 ,   11 8 ,   12 10 ,   13 12 ,   14 15 ,   15 5 ,   16 12 ,   17 8 ,   18 7 , 19
6 ,   20 6 ,   21 4 ,   22 5 ,   23 5 ,   24 3 ,   25 2 ,   26 7 ,   27 1 , 28 3 ,
29 3 ,   30 1 ,   31 2 ,   32 1 ,   33 2 ,   34 1 ,   35 2 ,   37 2 , 39 1 ,   40 2 ,
42 1 ,   43 2 ,   46 3 ,   50 1 ,   51 1 ,   52 3 ,   54 1 , 59 1 ,   61 1 ,   62 1 ,
63 2 ,   65 2 ,   73 1 ,   74 3 ,   75 2 ,   83 1 , 87 2 ,   92 1 ,   93 1 ,   97 1 ,
98 1 ,   102 1 ,   104 1 ,   107 1 ,   109 1 , 114 1 ,   127 1 ,   129 2 ,   130 1 ,
149 1 ,   158 1 ,   190 1 ,   201 1 , 277 1/;
* samp7
* /1 593 ,   2 195 ,   3 84 ,   4 75 ,   5 46 ,   6 41 ,   7 19 ,   8 16 ,   9 22 , 10
18 ,   11 8 ,   12 12 ,   13 13 ,   14 13 ,   15 10 ,   16 11 ,   17 11 , 18 3 ,   19
8 ,   20 3 ,   21 1 ,   22 5 ,   23 3 ,   24 3 ,   25 1 ,   26 1 , 27 4 ,   28 1 ,
```

```
29 5 ,   30 3 ,   32 1 ,   33 2 ,   34 3 ,   35 4 ,   37 1 , 39 1 ,   41 1 ,   42 3 ,
43 1 ,   46 2 ,   47 2 ,   48 3 ,   49 2 ,   52 1 , 54 1 ,   56 1 ,   58 1 ,   65 1 ,
71 1 ,   72 1 ,   73 2 ,   83 2 ,   85 2 , 86 1 ,   88 1 ,   89 1 ,   90 1 ,   95 1 ,
99 2 ,   105 1 ,   107 1 ,   110 2 , 111 1 ,   117 1 ,   134 1 ,   140 1 ,   146 1 ,
149 1 ,   152 1 ,   189 1 , 203 1 ,   307 1 /;
* samp8
* /1 602 ,   2 177 ,   3 90 ,   4 81 ,   5 43 ,   6 28 ,   7 22 ,   8 18 ,   9 19 , 10
16 ,   11 14 ,   12 8 ,   13 11 ,   14 7 ,   15 8 ,   16 14 ,   17 12 , 18 7 ,   19 6
,   20 6 ,   21 3 ,   22 6 ,   23 2 ,   24 2 ,   25 4 ,   26 7 , 27 4 ,   28 1 ,   29
1 ,   31 5 ,   33 1 ,   34 1 ,   35 1 ,   36 1 ,   37 2 , 38 1 ,   39 2 ,   41 3 ,
43 1 ,   44 3 ,   46 3 ,   48 1 ,   49 1 ,   50 1 , 51 1 ,   53 1 ,   55 1 ,   58 1 ,
59 1 ,   61 2 ,   64 1 ,   65 1 ,   69 1 , 71 1 ,   84 1 ,   88 1 ,   89 1 ,   91 4 ,
94 1 ,   99 2 ,   103 1 ,   106 2 , 112 1 ,   116 1 ,   118 1 ,   120 1 ,   125 1 ,
135 1 ,   138 1 ,   155 1 , 183 1 ,   204 1 ,   328 1/;
* samp9
* /1 621 ,   2 175 ,   3 78 ,   4 53 ,   5 51 ,   6 41 ,   7 29 ,   8 19 ,   9 10 , 10
17 ,   11 9 ,   12 15 ,   13 8 ,   14 12 ,   15 9 ,   16 12 ,   17 9 , 18 9 ,   19 8 ,
20 6 ,   21 7 ,   22 5 ,   23 5 ,   24 5 ,   25 3 ,   26 1 , 27 6 ,   28 1 ,   29 1 ,
30 1 ,   31 1 ,   32 1 ,   33 1 ,   34 3 ,   35 2 , 37 1 ,   38 1 ,   40 2 ,   41 1 ,
44 1 ,   45 1 ,   46 3 ,   47 1 ,   48 2 , 49 2 ,   50 1 ,   53 1 ,   54 1 ,   55 1 ,
60 1 ,   63 3 ,   64 1 ,   68 1 , 73 1 ,   77 1 ,   82 1 ,   83 1 ,   87 1 ,   91 1 ,
92 1 ,   93 1 ,   95 1 , 98 1 ,   101 2 ,   108 1 ,   115 2 ,   119 1 ,   122 1 ,
127 1 ,   130 1 , 134 1 ,   144 1 ,   150 1 ,   186 1 ,   194 1 ,   314 1 /;
* samp10
* /1 644 ,   2 179 ,   3 99 ,   4 56 ,   5 51 ,   6 33 ,   7 20 ,   8 19 ,   9 17 , 10
21 ,   11 13 ,   12 18 ,   13 13 ,   14 9 ,   15 6 ,   16 10 ,   17 8 , 18 6 ,   19 2
,   20 5 ,   21 7 ,   22 5 ,   23 3 ,   24 2 ,   25 4 ,   26 4 , 27 2 ,   28 2 ,   29
2 ,   30 2 ,   31 2 ,   32 2 ,   33 2 ,   34 2 ,   35 1 , 36 1 ,   38 1 ,   41 1 ,
42 1 ,   44 2 ,   47 4 ,   48 1 ,   50 2 ,   51 2 , 53 1 ,   54 2 ,   55 1 ,   56 1 ,
58 1 ,   61 1 ,   65 1 ,   67 1 ,   69 1 , 73 1 ,   80 3 ,   83 2 ,   86 1 ,   88 1 ,
90 1 ,   91 1 ,   93 1 ,   100 1 , 103 1 ,   104 1 ,   108 1 ,   109 1 ,   113 1 ,
116 1 ,   125 1 ,   129 1 , 140 1 ,   148 1 ,   163 1 ,   181 1 ,   191 1 ,   346 1 /;
* samp11,
* / 1 615 , 2 191 , 3 100 , 4 49 , 5 33 , 6 39 , 7 27 , 8 20 , 9 16 , 10 22 , 11 16 ,
12 11 , 13 11 , 14 9 , 15 8 , 16 8 , 17 8 , 18 9 , 19 9 , 20 3 , 21 2 , 22 3 , 23 6 ,
24 5 , 25 4 , 26 2 , 27 4 , 28 3 , 29 2 , 30 3 , 32 4 , 33 2 , 36 2 , 37 4 , 38 1 , 39
1 , 41 1 , 42 1 , 43 2 , 44 3 , 47 4 , 48 3 , 53 1 , 55 2 , 56 1 , 61 1 , 62 1 , 65 1
, 67 1 , 69 1 , 73 1 , 74 1 , 76 1 , 78 1 , 81 1 , 82 1 , 90 2 , 91 1 , 100 3 , 106 2
, 108 1 , 113 2 , 123 1 , 133 1 , 137 1 , 152 2 , 186 1 , 192 1 , 256 1 /;
* samp12
* / 1 618 , 2 185 , 3 106 , 4 53 , 5 41 , 6 37 , 7 19 , 8 20 , 9 17 , 10 14 , 11 14 ,
12 13 , 13 21 , 14 6 , 15 11 , 16 6 , 17 6 , 18 6 , 19 8 , 20 5 , 21 3 , 22 3 , 23 6 ,
24 3 , 25 5 , 26 4 , 27 4 , 28 2 , 29 3 , 30 4 , 31 2 , 32 2 , 33 1 , 34 1 , 35 2 , 38
1 , 42 3 , 44 2 , 45 1 , 46 2 , 47 2 , 49 1 , 50 1 , 52 2 , 53 1 , 54 1 , 58 3 , 70 2
, 75 1 , 76 3 , 78 1 , 79 1 , 82 2 , 91 1 , 96 1 , 100 2 , 101 1 , 105 1 , 108 1 , 111
1 , 116 1 , 122 1 , 127 1 , 131 1 , 142 1 , 143 1 , 148 1 , 154 1 , 170 1 , 191 1 ,
286 1 /;
* samp13
* /1 613 , 2 185 , 3 93 , 4 58 , 5 47 , 6 30 , 7 27 , 8 22 , 9 18 , 10 15 , 11 12 , 12
14 , 13 13 , 14 10 , 15 15 , 16 9 , 17 5 , 18 7 , 19 5 , 20 5 , 21 5 , 22 8 , 23 3 ,
24 4 , 25 2 , 26 3 , 27 4 , 28 2 , 29 7 , 30 1 , 32 1 , 33 2 , 34 2 , 37 2 , 39 2 , 41
1 , 43 3 , 44 1 , 46 3 , 48 1 , 49 2 , 51 1 , 56 1 , 59 2 , 63 1 , 64 1 , 65 1 , 66 1
, 68 2 , 71 1 , 73 1 , 76 1 , 85 1 , 89 1 , 90 2 , 91 1 , 92 2 , 98 1 , 99 1 , 104 1 ,
```

```
105 1 , 110 1 , 117 1 , 119 2 , 123 1 , 141 1 , 162 1 , 167 1 , 178 1 , 204 1 , 325 1
/;
* samp14
* / 1 619 , 2 166 , 3 102 , 4 59 , 5 51 , 6 31 , 7 22 , 8 23 , 9 15 , 10 17 , 11 14 ,
12 15 , 13 9 , 14 5 , 15 9 , 16 13 , 17 7 , 18 6 , 19 12 , 20 6 , 21 3 , 22 8 , 23 3 ,
24 1 , 25 2 , 26 4 , 27 1 , 28 2 , 29 3 , 30 2 , 31 2 , 32 2 , 33 1 , 34 1 , 36 2 , 37
2 , 38 2 , 39 1 , 40 2 , 41 2 , 43 1 , 45 2 , 48 1 , 50 1 , 51 4 , 56 2 , 58 1 , 59 2
, 60 1 , 62 1 , 64 1 , 66 1 , 78 1 , 79 1 , 80 1 , 84 1 , 85 1 , 88 1 , 89 1 , 91 2 ,
96 1 , 99 1 , 103 2 , 108 1 , 110 1 , 111 1 , 117 1 , 120 1 , 125 1 , 127 1 , 134 1 ,
143 1 , 151 1 , 175 1 , 186 1 , 300 1 /;
* samp15
* / 1 607 , 2 182 , 3 81 , 4 70 , 5 48 , 6 38 , 7 24 , 8 18 , 9 11 , 10 13 , 11 15 ,
12 15 , 13 5 , 14 9 , 15 9 , 16 13 , 17 12 , 18 4 , 19 8 , 20 3 , 21 4 , 22 4 , 23 3 ,
24 3 , 25 6 , 26 5 , 27 1 , 28 4 , 29 1 , 30 3 , 31 4 , 33 2 , 34 2 , 35 1 , 37 2 , 38
1 , 40 2 , 41 1 , 42 3 , 43 1 , 45 1 , 46 1 , 50 2 , 51 2 , 52 1 , 53 1 , 59 1 , 60 1
, 61 1 , 62 1 , 67 3 , 70 1 , 71 3 , 78 1 , 85 1 , 86 1 , 87 1 , 92 1 , 94 1 , 99 1 ,
100 1 , 105 1 , 109 1 , 110 2 , 111 1 , 115 1 , 116 1 , 119 1 , 128 1 , 130 1 , 146 1
, 156 1 , 174 1 , 199 1 , 307 1 /;
* samp16
* / 1 606 , 2 195 , 3 108 , 4 58 , 5 43 , 6 32 , 7 25 , 8 22 , 9 18 , 10 25 , 11 16 ,
12 9 , 13 6 , 14 6 , 15 8 , 16 9 , 17 7 , 18 10 , 19 4 , 20 6 , 21 9 , 22 3 , 23 4 ,
24 8 , 25 2 , 26 4 , 27 4 , 28 2 , 29 1 , 30 1 , 31 1 , 32 1 , 33 1 , 34 2 , 36 2 , 38
1 , 39 1 , 40 2 , 41 1 , 42 3 , 44 4 , 45 1 , 49 1 , 50 1 , 52 1 , 55 1 , 56 2 , 57 1
, 60 1 , 67 1 , 68 1 , 69 2 , 70 1 , 78 1 , 81 1 , 87 1 , 88 2 , 92 1 , 97 2 , 98 1 ,
101 1 , 104 1 , 106 1 , 113 1 , 116 1 , 119 1 , 121 1 , 124 2 , 135 1 , 138 1 , 146 1
, 190 1 , 223 1 , 328 1 /;
* samp17
* / 1 604 , 2 210 , 3 84 , 4 74 , 5 44 , 6 33 , 7 24 , 8 22 , 9 16 , 10 15 , 11 18 ,
12 13 , 13 10 , 14 7 , 15 7 , 16 12 , 17 4 , 18 9 , 19 7 , 20 6 , 21 3 , 22 6 , 23 4 ,
24 7 , 25 3 , 26 4 , 27 1 , 28 3 , 29 1 , 30 1 , 31 4 , 33 3 , 34 1 , 35 1 , 36 1 , 38
2 , 40 3 , 42 1 , 43 1 , 47 3 , 48 1 , 49 1 , 53 1 , 55 2 , 59 1 , 60 1 , 61 1 , 64 1
, 68 1 , 70 1 , 77 1 , 81 1 , 82 1 , 84 2 , 86 1 , 88 1 , 90 1 , 92 1 , 97 1 , 99 2 ,
106 1 , 107 1 , 108 1 , 109 1 , 116 1 , 122 1 , 123 1 , 128 1 , 131 1 , 134 1 , 145 1
, 146 1 , 174 1 , 212 1 , 290 1 /;
* samp18
* / 1 585 , 2 190 , 3 93 , 4 69 , 5 52 , 6 34 , 7 25 , 8 21 , 9 14 , 10 17 , 11 19 ,
12 11 , 13 9 , 14 12 , 15 6 , 16 9 , 17 9 , 18 6 , 19 10 , 20 3 , 21 4 , 22 4 , 23 3 ,
24 5 , 25 4 , 26 1 , 27 4 , 28 5 , 29 1 , 30 4 , 31 2 , 32 1 , 33 2 , 34 1 , 36 1 , 38
1 , 39 3 , 40 1 , 42 4 , 43 1 , 47 2 , 48 2 , 51 1 , 53 1 , 54 1 , 55 1 , 57 1 , 61 3
, 64 1 , 69 1 , 71 1 , 74 1 , 75 1 , 76 1 , 77 1 , 79 1 , 83 1 , 88 1 , 89 1 , 90 1 ,
93 1 , 94 1 , 99 1 , 102 1 , 106 1 , 112 2 , 117 2 , 118 1 , 129 1 , 132 2 , 151 1 ,
167 1 , 210 1 , 313 1 /;
* samp19
* / 1 614 , 2 189 , 3 105 , 4 59 , 5 41 , 6 44 , 7 30 , 8 25 , 9 14 , 10 12 , 11 11 ,
12 8 , 13 13 , 14 11 , 15 6 , 16 10 , 17 6 , 18 8 , 19 6 , 20 1 , 21 10 , 22 2 , 23 7
, 24 7 , 25 2 , 26 5 , 27 2 , 28 2 , 29 2 , 30 3 , 31 1 , 33 2 , 35 1 , 36 1 , 37 6 ,
38 1 , 39 1 , 40 2 , 41 1 , 42 2 , 46 2 , 48 1 , 49 1 , 50 1 , 51 2 , 52 2 , 55 1 , 60
1 , 62 2 , 63 1 , 66 1 , 70 1 , 73 1 , 77 1 , 80 2 , 85 2 , 86 1 , 87 1 , 99 1 , 101 1
, 106 1 , 107 1 , 109 1 , 110 1 , 111 1 , 112 1 , 113 1 , 121 1 , 128 1 , 146 1 , 154
1 , 162 1 , 175 1 , 188 1 , 287 1 /;
* samp20
* / 1 628 , 2 192 , 3 96 , 4 62 , 5 46 , 6 29 , 7 26 , 8 18 , 9 20 , 10 18 , 11 15 ,
12 12 , 13 4 , 14 16 , 15 12 , 16 10 , 17 7 , 18 5 , 19 6 , 20 5 , 21 5 , 22 4 , 23 4
, 24 7 , 25 4 , 26 3 , 27 1 , 28 2 , 29 1 , 30 1 , 31 2 , 32 1 , 33 3 , 34 1 , 36 1 ,
```

```
38 4 , 39 2 , 40 2 , 42 1 , 43 1 , 44 2 , 45 3 , 46 2 , 47 1 , 49 1 , 52 1 , 53 1 , 54
1 , 56 1 , 57 1 , 63 1 , 66 1 , 68 1 , 69 1 , 77 3 , 78 1 , 80 1 , 82 1 , 88 1 , 90 1
, 92 2 , 93 1 , 99 1 , 101 2 , 104 1 , 105 1 , 110 1 , 115 1 , 130 1 , 142 1 , 145 1 ,
147 2 , 183 1 , 187 1 , 288 1 /;

Parameter Np  size of the population /56372/;
          display Np;

Parameter Ns size of sample ;
         Ns = sum(j,ord(j)*m(j));
* round just in case m(j) are not integers (perfect samples)
         Ns = round(Ns);
         display Ns;

Parameters
         prob(j,i)  prob size j in the sample given size i in the population;

Parameter thresh /1.E-6/ ;
* loop to compute prob(j,i)

prob('1','1' ) = Ns/Np;
Loop(i $( ord(i) ge 2 ),
  prob('1', i) = prob('1', i-1) * ord(i) * (Np - Ns  -
            ord(i) +2)/(ord(i)-1)/(Np-ord(i) +1) ;
  if (prob('1',i)< thresh, prob('1',i)=0;);
 Loop(j $(ord(j) le ( ord(i)-1 ) ),
      prob(j+1, i) = prob(j, i) * ( ord(i) - ord(j) ) *
                (Ns - ord(j) )/( ord(j) + 1)/(Np - Ns - ord(i) + ord(j) +1);
  if (prob(j+1,i)< thresh, prob(j+1,i)=0 ;) ;
      );
);

option decimals = 6;

Display prob;
Variables
   classes     estimate of total number of classes in population (for reporting only)
   mu(i)       num classes in pop of size i
   Em(j)       Expected m(j) from Goodman's equations
   pdev(i)     positive deviation vars in estimate eqns
   ndev(i)     negative deviation vars in estimate eqns
   maxerror    largest absolute error in estimate eqns

   objvar1       objective variable for L1 norm
   objvar2       objective variable for L2 norm
   objvarmax     objective variable for Lmax norm
     ;

Positive variables  mu, classes, Em, pdev, ndev;

* set upper bounds

   mu.up(i) = Np;
   mu.l(i) = 1;
```

```
   pdev.up(i) = 0.01*Np;
   ndev.up(i) = 0.01*Np;
   Em.up(j) = Ns ;
   Em.lo(j) = 1.E-10;
   Em.l(j)=max(Em.lo(j),m(j));
   maxerror.up = Np ;

Equations
   defineEm(j)    define Em(j)
   Estimate(i)    compute absolute error in estimate equations
   Estimatelo(i)  lower limit on error in estimate equations
   Estimateup(i)  upper limit on error in estimate equations
   Population     should sum to population size
   Summu(j)       classes in sample arise from same size or larger classes in
population
   Total          calculate the number of classes (reporting only)
   mono           monotonicity constraint
   mu1max         ensures that mu1 is the largest
   Obj1           sum of absolute error
   Obj2           sum of squares of errors
   Objmax         max error
   ;

   defineEm(j)$m(j)..  Em(j) =e= sum(k$(ord(k) ge ord(j)),mu(k)*prob(j,k)) ;

   Population      ..  sum(i,ord(i)*mu(i)) =e= Np ;


   Summu(j)$m(j)   ..  sum(i $ (ord(i) ge ord(j)),mu(i)) =g= m(j) ;

   Total           ..  sum(i,mu(i)) =e= classes;

   estimate(i)     ..  mu(i)* sum(j $ (ord(j) le ord(i)),prob(j,i)*(1-(m(j)/Em(j))))
                       =e= pdev(i)-ndev(i);

   estimatelo(i)   ..  mu(i)* sum(j $ (ord(j) le ord(i)),prob(j,i)*(1-(m(j)/Em(j))))
                       =g= -maxerror ;

   estimateup(i)   ..  mu(i)* sum(j $ (ord(j) le ord(i)),prob(j,i)*(1-(m(j)/Em(j))))
                       =l= maxerror ;

   mono(i)         ..  mu(i) =g= mu(i+1);

   mu1max(i)       ..  mu('1') =g= mu(i);

   Obj1            ..  objvar1 =e= sum(i,(pdev(i)+ndev(i))) ;

   Obj2            ..  objvar2 =e= sum(i,sqr(mu(i))*sqr(sum(j$(ord(j) le ord(i)),
                         prob(j,i)*(1-(m(j)/Em(j)))))) ;
   Objmax          ..  objvarmax =e= maxerror ;

Model L1 /
defineeM, population,summu,total, estimate,
*mu1max,
```

```
mono, obj1 /;
*L1.optfile=1;
*Model L2 /
*defineeM,
*population,summu,total,

*mono,
*obj2 /;
*Model Lmax /
*defineeM,
*population,summu,total,
*estimatelo,estimateup,
*mono,
*objmax /;
L1.optfile = 1;
*L2.optfile = 1;
*Lmax.optfile = 1;

option nlp = oqnlp;
option decimals=6;

* solve L1 problem

Solve  L1 using nlp minimizing objvar1;
Display classes.l, objvar1.l, mu.l;

* compute and display errors in estimator eqns
*Parameters esterr(i);
*  esterr(i) =mu.l(i)* sum(j $ (ord(j) le ord(i)),prob(j,i)*(1-(m(j)/Em.l(j))));
*Display esterr ;

* solve L2 problem

*Solve  L2 using nlp minimizing objvar2;
*Display classes.l, objvar2.l,
*poperr.l,mu1err.l,
*mu.l ;
* compute and display errors in estimator eqns
*Parameters esterr(i);
*  esterr(i) =mu.l(i)* sum(j $ (ord(j) le ord(i)),prob(j,i)*(1-(m(j)/Em.l(j))));
*Display esterr ;

* solve Lmax problem

*Solve  Lmax using nlp minimizing objvarmax;
*Display classes.l, maxerror.l,poperr.l,mu1err.l,mu.l ;
* compute and display errors in estimator eqns
*Parameters esterr(i);
*  esterr(i) =mu.l(i)* sum(j $ (ord(j) le ord(i)),prob(j,i)*(1-(m(j)/Em.l(j))));
*Display esterr ;
```

Appendix B - Second population

fcount	COUNT	PERCENT
1	1175	40.8838
2	410	14.2658
3	225	7.8288
4	128	4.4537
5	101	3.5143
6	69	2.4008
7	56	1.9485
8	52	1.8093
9	36	1.2526
10	37	1.2874
11	28	0.9743
12	28	0.9743
13	16	0.5567
14	19	0.6611
15	18	0.6263
16	22	0.7655
17	13	0.4523
18	13	0.4523
19	16	0.5567
20	9	0.3132
21	11	0.3827
22	15	0.5219
23	13	0.4523
24	8	0.2784
25	12	0.4175
26	5	0.1740
27	10	0.3479
28	5	0.1740
29	4	0.1392
30	9	0.3132
31	7	0.2436
32	7	0.2436
33	6	0.2088
34	8	0.2784
35	5	0.1740
36	2	0.0696
37	7	0.2436
38	7	0.2436
39	6	0.2088
40	5	0.1740
41	6	0.2088
42	4	0.1392
43	7	0.2436
44	2	0.0696
45	6	0.2088
46	7	0.2436
47	5	0.1740
48	3	0.1044
50	3	0.1044
51	3	0.1044
52	3	0.1044

53	1	0.0348
54	2	0.0696
55	2	0.0696
56	3	0.1044
57	1	0.0348
58	2	0.0696
59	2	0.0696
60	2	0.0696
61	2	0.0696
62	2	0.0696
63	1	0.0348
65	5	0.1740
66	2	0.0696
67	1	0.0348
68	1	0.0348
69	3	0.1044
70	1	0.0348
71	1	0.0348
73	4	0.1392
75	2	0.0696
76	2	0.0696
77	4	0.1392
78	1	0.0348
80	2	0.0696
81	1	0.0348
82	3	0.1044
83	2	0.0696
84	3	0.1044
85	3	0.1044
86	1	0.0348
87	8	0.2784
88	1	0.0348
90	1	0.0348
91	1	0.0348
93	5	0.1740
94	1	0.0348
95	3	0.1044
96	1	0.0348
97	1	0.0348
99	1	0.0348
100	1	0.0348
101	1	0.0348
102	1	0.0348
103	1	0.0348
104	1	0.0348
108	2	0.0696
109	1	0.0348
110	2	0.0696
112	3	0.1044
113	1	0.0348
114	1	0.0348
115	1	0.0348
116	2	0.0696
117	2	0.0696

118	1	0.0348
120	2	0.0696
122	3	0.1044
123	2	0.0696
124	1	0.0348
128	2	0.0696
129	1	0.0348
131	1	0.0348
133	1	0.0348
136	2	0.0696
137	2	0.0696
141	1	0.0348
144	1	0.0348
145	1	0.0348
149	1	0.0348
152	1	0.0348
154	1	0.0348
156	1	0.0348
158	1	0.0348
159	2	0.0696
161	1	0.0348
163	1	0.0348
164	1	0.0348
165	1	0.0348
166	2	0.0696
169	2	0.0696
170	2	0.0696
174	1	0.0348
180	1	0.0348
185	1	0.0348
190	1	0.0348
196	1	0.0348
202	1	0.0348
216	1	0.0348
218	1	0.0348
228	1	0.0348
232	1	0.0348
240	2	0.0696
247	1	0.0348
250	1	0.0348
251	1	0.0348
254	1	0.0348
269	1	0.0348
272	1	0.0348
276	1	0.0348
278	1	0.0348
280	1	0.0348
291	1	0.0348
293	1	0.0348
307	1	0.0348
314	1	0.0348
317	1	0.0348
321	1	0.0348
343	1	0.0348

350	1	0.0348
362	1	0.0348
394	1	0.0348
413	1	0.0348
438	1	0.0348
454	1	0.0348
466	2	0.0696
483	1	0.0348
494	1	0.0348
515	1	0.0348
529	1	0.0348
530	1	0.0348
533	1	0.0348
551	1	0.0348
597	1	0.0348
617	1	0.0348
628	1	0.0348
639	1	0.0348
646	1	0.0348
672	1	0.0348
691	1	0.0348
707	1	0.0348
717	1	0.0348
771	1	0.0348
820	1	0.0348
855	1	0.0348
890	1	0.0348
1092	1	0.0348
1153	1	0.0348
1835	1	0.0348

Appendix C – OQNLP.OPT File

```
************* OQMS Options File **************
*
* All records have the free format form <keyword>  <value>
* (d) means default choice
* an asterisk in column one means the line is ignored
* nothing is case sensitive
* a 1 line oqms.opt file with a single line containing the word
* help will cause a template options file to be written to the
* GAMS log file.
*
*
*help
echo
*debug
*bad_option
*
* STARTING POINT GENERATOR
*POINT_GENERATION        random
POINT_GENERATION        optquest

*
*               RESOURCE LIMITS

* maximum runtime in seconds (1000(d))
MAXTIME                       20000

*  maximum number of nlp solver calls (1000(d))
MAX_SOLVER_CALLS               1000

*  maximum number of local optima found(1000(d))
MAX_LOCALS                     1000

* total number of OQMS iterations: (1000(d))
ITERATION_LIMIT                 1000
*
*               ALGORITHM PARAMETERS AND OPTIONS

* nlp solver to be used (conopt,lsgrg(d),minos,snopt)
NLPSOLVER                      conopt.1


* start with call to local solver before beginning stage one
START_WITH_NLPSOLVER                   1

* all OptQuest trial points satisfy linear constraints (0-no(d),1-yes)
USE_LINEAR_CONSTRAINTS:             0
* use merit filter to determine if NLP solver should be started
* (0-no,1-yes(d))
USE_MERIT_FILTER                    0

* use distance filter to determine if NLP solver should be started
* (0-no,1-yes(d))
```

```
USE_DISTANCE_FILTER                0

* dynamic merit filter, which may increase THRESHOLD_INCREASE_FACTOR if the merit
* filter rejects WAITCYCLE consecutive points (0-no,1-yes(d))
DYNAMIC_MERIT_FILTER                    0

*  dynamic distance filter logic which reduces the basin radius if it rejects
*  trial points for WAITCYCLE consecutive cycles (0-no,1-yes(d))
DYNAMIC_DISTANCE_FILTER                 0

*  distance filter logic which reduces pairs of radii if they overlap
* (0-no,1-yes(d))
BASIN_OVERLAP_FIX                       0

* selects different options for the length of stage 1. Values are:
* 0:use STAGE1_ITERATIONS iterations (d)
* k,k>=1:use k "generations" of OptQuest trial points, where each
* generation is generated by the initial population (k=1),
* the population after 1 update (k=2), etc
INIT_CYCLE                              0

* number of initial optquest iterations in stage 1 (200(d))
STAGE1_ITERATIONS                  200

* value used as upper/lower bound on any variable with no bound
* (1.e5(d))
ARTIFICIAL_BOUND                     1.e4

* penalty fcn used in stage 1:
* (0-OptQuest's, 1-exact penalty)(0(d))
PENALTY_FUNCTION                        0

* starting value for Lagrange multipliers (1000(d))
STARTING_MULTIPLIER                   1000

* number of iterations before merit filter threshold is increased (20(d))
MERIT_WAITCYCLE                            20

* number of iterations before distance filter radius is reduced (20(d))
* used only if DYNAMIC_DISTANCE_FILTER is 1
DISTANCE_WAITCYCLE                           20


* factor to increase merit filter threshold (0.2(d))
THRESHOLD_INCREASE_FACTOR              0.2

* factor to decrease distance filter basin size(0.2(d))
BASIN_DECREASE_FACTOR                  0.2

* factor used in distance filter:don't start NLP solver at a trial point if
* distance of trial point from any local optimum found so far <=
* distance_factor*(largest distance traveled
* to get to that local optimum)(0.75(d))
DISTANCE_FACTOR                        0.75
```

```
* Option MAX_SOLVER_CALLS_NOIMPROVEMENT  controls algorithm termination
* based on lack of progress in locating better local solutions.
* if k solver calls are executed (and terminate with feasible solutions)
* without detecting a local solution with an improved objectve value,
* the algorithm will terminate.  The default value for
* MAX_SOLVER_CALLS_NOIMPROVEMENT is  10% of the value of MAX_SOLVER_CALLS.
* Setting MAX_SOLVER_CALLS_NOIMPROVEMENT to 0 invokes that default.
* Setting MAX_SOLVER_CALLS_NOIMPROVEMENT to k sets the count to k as described
*   above.
* Setting MAX_SOLVER_CALLS_NOIMPROVEMENT to -1 (or any negative value)
*   removes this count as a termination criterion
MAX_SOLVER_CALLS_NOIMPROVEMENT                          1000

* insert nlp solver log into gams log (only applicable under gams)
SOLVER_LOG_TO_GAMS_LOG                0

********** OPTQUEST parameters

* in MINLP's, OptQuest knows only about discrete variables
*(0-no: OptQuest changes both continuous and discrete variables,
* 1-yes: OptQuest changes only discrete variables (1(d))
DISCRETES_ONLY_OPTQUEST:                       1

* use only Optquest iterations (0-no(d), 1-yes,no solver calls are made):
OPTQUEST_ONLY:                        0

* OptQuest population size (10(d)):
OQ_POPSIZE:                           10

* OptQuest Search Strategy: aggressive, boundary(d), crossover
SEARCH_TYPE:                          boundary

* OQ search strategy parameter (0.5(d))
SEARCH_PARAMETER:                     0.5

*                      OUTPUT CONTROL

* setting the frequency of iteration print to OQNLP log file (20(d))
LOGFILE_ITN_PRINT_FREQUENCY             20

* setting the frequency of iteration print to GAMS log file (20(d))
GAMS_ITN_PRINT_FREQUENCY                20

* set OQNLP print level (0(d))
* 0 gives least output
* any positive value gives more output
OQNLP_PRINT_LEVEL                     0

* turn on different level of debug output (0-none,1-more,2-even more)
* output goes to a file (in the gams project directory if you are using gams)
* called <modelname>.dbg
OQMS_DEBUG                             0
```

```
*  turn on iteration log and termination messages to screen (0-no,1-yes(d))
ENABLE_SCREEN_OUTPUT                    1

* specify path and name for file to which all local solutions
* found will be written
LOCALS_FILE                          locals.out

*  specify format for locals file 'report' = human readable
*    'data1' =  <index of local optimum> <objval> <var index> <var value>
LOCALS_FILE_FORMAT                     data1
*enable stats log
ENABLE_STATISTICS_LOG                   1


********** LSGRG output options
*(all 0 defaults except 1 for disable error messages)
*
* initial printing
GRG_INPRNT:                            0
* final printing
GRG_OTPRNT:                            0
* printlevel set
GRG_PRINTLEVEL:                        0
* supress any error messages coming out of lsgrg2  0,1
GRG_DISABLE_ERROR_MSGS                 1
*
```

Appendix D – CONOPT.OPT File

```
* Set option RTMAXJ larger than the defalut of 1.0E5 as CONOPT fails
* if it finds a Jacobian element larger than RTMAXJ.  1.0E5 is way
* too small.
*
 RTMAXJ 1.0E8
 LFILOG    1
```