ALGORITHM 616
Fast Computation of the Hodges-Lehmann Location Estimator

JOHN F. MONAHAN
North Carolina State University

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*sorting and searching*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*statistical computing*

General Terms: Algorithms, Design

Additional Key Words and Phrases: Computational complexity, Hodges-Lehmann location estimator, divide and conquer, quicksort, median

DESCRIPTION

HLQEST is a FORTRAN function subprogram for computing the Hodges-Lehmann [5] estimator

$$\hat{\mu} \equiv \text{median}\left\{\frac{(X_i + X_j)}{2}, 1 \le i \le j \le n\right\}.^1 \qquad (1)$$

This robust and highly efficient estimator [2] has not been widely used by statisticians because its apparent time computational complexity is $O(n^2 \log n)$. Improvements in computing $\hat{\mu}$ have previously been made with an iterative algorithm [11] and with some fast theoretical techniques [7, 8]. HLQEST is exact and fast, with expected time complexity of $O(n \log n)$.

The estimator $\hat{\mu}$ arises from inverting the one-sample Wilcoxon test statistic. That is, $\hat{\mu}$ is a root of

$$O = W(\mu) = \sum_{i=1}^{n} \text{rank}(|X_i - \mu|) \times \text{sign}(X_i - \mu), \qquad (2)$$

where $W(\mu)$ is the Wilcoxon test statistic for the hypothesis $H{:}E(X_i) = \mu$. Notice

[1] The median for an even number of values is always taken to be the average of the two middle values.

Received 10 December 1981; revised 20 May 1983; accepted 2 December 1983
Author's address; Department of Statistics, North Carolina State University, Raleigh, NC 27650.

that $W(\mu)$ is a monotone step function and can yield multiple roots; taking the midrange of the roots yields a unique estimator and the definition (1).

McKean and Ryan [11] used the representation of $\hat{\mu}$ as a root of (2) as the basis for their algorithm, recommending the Illinois algorithm, (see [3], pp. 231–232), a variant of regula falsi, for finding the root iteratively. When there is an interval of roots, the two endpoints, roots of $W(\mu) = \pm\epsilon$, must be found; otherwise the root of (2) can differ substantially from definition (1). Note that computing $\hat{\mu}$ as a single root of (2) using regula falsi is the reported method in the so-called "Princeton study" [1]. Finally while McKean and Ryan deal with the two-sample Hodges-Lehmann estimator, the one-sample problem is very similar.

Johnson and Kashdan [7] and Johnson and Mizoguchi [8] produced "fast" algorithms for selection from multisets, for which $\hat{\mu}$ is a special case. The corresponding two-sample problem is analyzed by Johnson and Ryan [9]. However, no implementation of a fast exact algorithm is extant.

The exact algorithms of [7] and [8] and those that follow are based on the "divide and conquer" theme. The unique feature of the problem is that the structure allows the partitioning to be done in $O(n)$ time, while there are $O(n^2)$ elements. First of all, the values are to be sorted so that values of X_i appear in nondecreasing order. By placing the sum $X_i + X_j$ in the (i, j)th element of an upper triangular matrix, the number of elements less than some number a can be found by starting at the upper right corner and moving to the diagonal. To keep track of what elements are between two numbers a and b, only pointers to the first and last elements in each row are needed. Of course, this matrix is never formed. All of these algorithms for finding the kth smallest what follow the structure

$S_0 = \{(X_i + X_j), \quad 1 \le i \le j \le n\}; \qquad m = 0;$
while it's a good idea *do*
  Find a partition element a_m;
  Let L_m be the elements of S_0 that are less than a_m;
  if $|L_m| > k$ *then* $S_{m+1} = S_m \cap L_m$
    else $S_{m+1} = S_m \cap L_m^c$;
end while;

For the exact algorithms of [7] and [8], "it's a good idea" means that $|S_m| > n$, otherwise the job is completed directly by sorting S_m. Also, the partition element a_m is chosen to be the weighted median of row medians of elements S_m where the weight is the number in the row in S_m. Under this scheme, a_m cuts off at least one fourth of the elements at every step,

$$|S_{m+1}| \le 3/4\ |S_m|, \tag{3}$$

so that the number of steps is $O(n \log n)$. Since the initial sorting takes $O(n \log n)$ and the weighted medians can be found using a fast $O(n)$ median routine, the total time complexity is $O(n \log n)$.

As implemented by the author in algorithm HLFEST, some changes are necessary. First, S_m must be split into three pieces: $<a_m$, $=a_m$, and $>a_m$, as recommended [8], in order to handle troublesome ties. Second, the fast median algorithm is impractical for most sample sizes encountered in practice; sorting was used to handle this subproblem, increasing the complexity to $O(n \log^2 n)$.

HLFEST proved to be fast enough for some recent Monte Carlo work by the author [12], but improvements were sought. It was believed that too much time was spent in finding the partition element a_m.

The QUICKSORT and FIND algorithms of Hoare [4] suggest an alternative: choose a_m at random from S_m. The second improvement is to stop the process when the largest or smallest element of S_{m+1} is sought. That is, "it's a good idea" is changed to

$$k \neq |L_m| \qquad \text{or} \qquad k \neq |L_m| + 1. \tag{4}$$

Also, "Find a partition element a_m;" is translated to

$$\text{Randomly choose an element in } S_m \text{ (all equally likely).} \tag{5}$$

Ties again present a problem: if $S_m = S_{m-1}$ then ties are suspected and a_m is replaced by the midrange of S_m, unless max S_m = min S_m where the process is stopped.

These changes were implemented in a subroutine called HLQIST. To analyze its complexity, we need only consider the random value of m when it leaves the *while* loop. Let $M_{k,l}$ be the expected number of steps in the *while* loop for finding the kth smallest of l elements. Since the max or min of S_m can be found in one step ($M_{1,l} = M_{l,l} \equiv 1$) costing $O(n)$, then for $1 < k < l$

$$\begin{aligned} M_{k,l} &= 1 + l^{-1}\left(\sum_{i=1}^{k} M_{k-i+1,l-i+1} + \sum_{i=k+1}^{l} M_{k,i-1}\right) \\ &= \left(2 + l + \sum_{i=1}^{k-2} M_{k-i,l-i} + \sum_{i=k+1}^{l-1} M_{k,i}\right) \Big/ (l-1). \end{aligned} \tag{6}$$

The recurrence relationship (6) can be analyzed to show that

$$M_{k,l} \leq a[\log k + \log(l - k + 1)] + b, \tag{7}$$

where a is unity and $b \approx 1.2$. It can be easily shown that $M_{2,l} = M_{l-1,l} = H_{l-1} + 1$, where H_i is the ith harmonic number. Since the initial sorting requires $O(n \log n)$ it is only necessary that $M_{k,n} = O(\log n)$ so that the total complexity of HLQIST is $O(n \log n)$.

Again, improvements were sought and two are implemented in the final algorithm HLQEST. First, a_0 is chosen to be 2 times the median $\{X_i\}$. Second, a_m is subsequently chosen as a random row median, where the probabilities are proportional to the number in that row that are in S_m, analogous to the original scheme. Both adjustments prove useful. The complexity of HLQEST is impossible to analyze, but should not change from $O(n \log n)$.

Notice that when n mod 4 is 0 or 3 then the number of values in S_0 is even and so the middle values must be found and averaged. As a consequence, all of these algorithms are designed to find one or two consecutive order statistics from S_0. They can easily be adjusted to find any consecutive pair of order statistics, as for constructing confidence intervals.

To compare the performance of these algorithms in practice, their FORTRAN implementations were timed using the IBM 3081 at the Triangle Universities Computation Center. For each of seven sample sizes, the average of 5 sets of 100

Table I. Time in Milliseconds/Sample

		Method					
		HLFEST	HLQIST	HLQEST	Sort	HDGSL1	HDGSL2
Normal n =	5	0.34	0.26	0.20	0.20	0.42	0.44
	10	0.80	0.64	0.50	0.86	0.64	0.68
	20	1.98	1.48	1.30	3.80	0.72	0.72
	50	6.46	4.24	3.62	29.00	2.38	2.78
	100	15.42	9.74	8.64	—	3.90	3.94
	200	36.66	21.96	19.24	—	8.20	8.28
	1000	257.04	140.54	122.54	—	44.84	49.92
Uniform n =	100	14.96	8.74	7.56	—	2.90	2.98
	1000	257.48	130.10	111.08	—	34.68	40.48
Discrete n =	100	9.96	6.34	5.30	—	5.70	5.70
	1000	130.32	68.82	57.32	—	62.74	63.82
Length of Compiled Code		1544 +(622)[a]	1620 +(378)[b]	1688 +(378)[b]	—	696 +(472)[c]	696 +(472)[c]

NOTE: The length of code is measured in bytes and does not include a sorting subroutine.
Auxiliary function subprograms
[a] Finds weighted median.
[b] Pseudorandom number generator [13].
[c] Evaluates $W(\mu)$ [1].

replications each are given in Table I. These values give the average computing time for each algorithm and sample size pair in terms of milliseconds per sample. Each sample was composed of *iid* uniform or standard normal pseudorandom variables obtained from the IMSL [6] routines GGUBS, an implementation of the Lewis, Goodman and Miller algorithm [10], and GGNPM. Samples labelled "Discrete" in the table are integer parts of five times the normal samples. Within HLQIST and HLQEST, Schrage's [13] portable FORTRAN implementation of the same algorithm [10] was the source of random variables. The label "Sort" refers to the straightforward method of creating the $n(n+1)/2$ pairs $(X_i + X_j)$ and sorting. Algorithms HDGSL1 and HDGSL2 are streamlined versions of the iterative methods described earlier, using regula falsi and the Illinois method, respectively, to find only a single root of $W(\mu) = 0$. When n mod 4 is 0 or 3, the additional effort for finding two roots (as in [11]) instead of one (as in [1]) would depend greatly on the sophistication of the algorithm.

From Table I, notice that HLQEST is superior to any other exact method and better than the two iterative methods in small sample sizes. In large samples, the error tolerance and sophistication of iterative methods, as well as the type of data, will determine the best method.

APPENDIX. COMPLEXITY OF HLQIST

We will follow an induction argument, using (6)

$$(l-1)M_{k,l} = l + 2 + \sum_{i=1}^{k-2} M_{k-i,l-i} + \sum_{i=k+1}^{l-1} M_{k,i}. \tag{A1}$$

For $1 < k < l \leq t$ (for some $t > 2$), it can be shown that

$$M_{k,l} \leq b + a \log k + a \log(l - k + 1). \tag{A2}$$

Therefore, using (A1) we can show that

$$(l-1)M_{k,l} \le (l+2) + (l-3)b + a(l-3/2)[\log k + \log(l-k+1)] - a(l-1). \tag{A3}$$

Therefore, if $a = 1$, then (A2) is true for $t + 1$ and thus all $1 < k < l$, as long as b is chosen so that (A2) is true for some $t \ge l$, for which $b = 1.2$ is sufficient for $t = 6$. Note that the logarithms used here are natural logarithms, arising from an inequality from Stirling's approximation:

$$\log(k-1)! \le (k-1/2)\log k - k + 1. \tag{A4}$$

ACKNOWLEDGMENTS

The author wishes to thank Dennis D. Boos and Carla D. Savage for their helpful suggestions, advice and encouragement. Useful comments from the editor and referees are appreciated.

REFERENCES

1 ANDREWS, D.F., BICKEL, P.J., HAMPEL, F.R., HUBER, P.J., ROGERS, W.J., AND TUKEY, J.W. *Robust Estimators of Location Survey and Advances* Princeton Univ. Press, Princeton, N.J , 1972.
2. BICKEL, P J. On some robust estimates of location. *Ann Math. Statist. 36* (1965), 847-858.
3. DAHLQUIST, G AND BJORK, A. *Numerical Methods.* Prentice-Hall, Englewood Cliffs, N.J., 1974.
4. HOARE, C A R. Algorithm 63 (PARTITION) and Algorithm 65 (FIND). *Commun. ACM 4*, 7 (July 1961), 321-322.
5. HODGES, J.L , AND LEHMANN, E L Estimates of location based on rank tests. *Ann. Math. Statist 34* (1963), 598-611.
6. INTERNATIONAL MATHEMATICS AND STATISTICAL LIBRARIES, INCORPORATED. *The IMSL Library, Vol.* 2, 8th ed., IMSL, Houston, Tex., 1980.
7. JOHNSON, D.B., AND KASHDAN, S.D. Lower bounds for selection in $X + Y$ and other multisets. *J ACM, 25*, 4 (Oct. 1978), 556-570.
8 JOHNSON, D.B., AND MIZOGUCHI, T. Selecting the kth element in $X + Y$ and $X_1 + X_2 + \cdots + X_m$. *SIAM J. Comput 7* (1978), 147-153
9 JOHNSON, D.B , AND RYAN, T.A. Fast computation of the Hodges-Lehmann estimator—theory and practice. In *Proc Statist Comp. Sec ASA* (1978), 1-2.
10. LEWIS, P A.W., GOODMAN, A.S. AND MILLER, J.M. A pseudorandom number generator for the System/360. *IBM Syst J 8* (1969), 136-146.
11. MCKEAN, J.W., AND RYAN, T.A. Algorithm 516. An algorithm for obtaining confidence intervals and point estimates based on ranks in a two-sample location problem. *ACM Trans. Math. Softw.* 3, 2 (June 1977), 183-185.
12. MONAHAN, J.F. The sampling densities of some robust estimators. Presented at the Pacific Area Statistical Conference (Tokyo, Dec. 15-17, 1982), Pacific Statistical Institute. Proceedings to be published.
13. SCHRAGE, L. A more portable Fortran random number generator. *ACM Trans Math. Softw 5*, 2 (June 1979), 132-138.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service (see page 355 for order form).]

```
      REAL FUNCTION HLQEST(X, N, LB, RB, Q)                          HLQ  10
C                                                                    HLQ  20
```

```
C     REAL FUNCTION HLQEST
C
C     PURPOSE         COMPUTES THE HODGES-LEHMANN LOCATION ESTIMATOR:
C                     MEDIAN OF ( X(I) + X(J) ) / 2   FOR 1 LE I LE J LE N
C
C     USAGE           RESULT = HLQEST(X,N,LB,RB,Q)
C
C     ARGUMENTS  X    REAL ARRAY OF OBSERVATIONS  (INPUT)
C                   * VALUES OF X MUST BE IN NONDECREASING ORDER *
C
C                N    INTEGER NUMBER OF OBSERVATIONS  (INPUT)
C                   * N MUST NOT BE LESS THAN 1 *
C
C                LB   INTEGER ARRAY OF LENGTH N FOR WORKSPACE
C
C                RB   INTEGER ARRAY OF LENGTH N FOR WORKSPACE
C
C                Q    INTEGER ARRAY OF LENGTH N FOR WORKSPACE
C
C          NOTE ---   ONLY LB,RB, AND Q ARE CHANGED IN COMPUTATION
C
C    EXTERNAL ROUTINE
C               RAN   FUNCTION PROVIDING UNIFORM RANDOM VARIABLES
C                     IN THE INTERVAL (0,1)
C                     RAN REQUIRES A DUMMY INTEGER ARGUMENT
C
C    NOTES            HLQEST HAS AN EXPECTED TIME COMPLEXITY ON
C                     THE ORDER OF N * LG( N )
C
```