```
C
C       CALCULATE SPECTRAL ESTIMATES FOR K = NF
C
      N2 = N2 + 1
      NF = K + 1
      SM = FLOAT(MOD(NT, 2) + 1) / 2.0
      SX(NF) = SM * SQCABS(ZX(NC))
      SY(NF) = SM * SQCABS(ZY(NC))
      ZYX(NF) = SM * ZY(NC) * CONJG(ZX(NC))
      IF (L .EQ. 0) GOTO 150
      DO 125 I = N2, NN
      SX(NF) = SX(NF) + SQCABS(ZX(I))
      SY(NF) = SY(NF) + SQCABS(ZY(I))
      ZYX(NF) = ZYX(NF) + ZY(I) * CONJG(ZX(I))
  125 CONTINUE
  150 L = L + 1
      SM = 1.0 / FLOAT(NT * L)
      SX(NF) = SM * SX(NF)
      SY(NF) = SM * SY(NF)
      ZYX(NF) = SM * ZYX(NF)
      RETURN
      END
```

Algorithm AS 74

L1-norm Fit of a Straight Line

By A. N. SADOVSKI†

Rothamsted Experimental Station

Keywords: L1-NORM; LINEAR REGRESSION

LANGUAGE

ISO Fortran

DESCRIPTION AND PURPOSE

The algorithm fits the simple linear form

$$y = \alpha + \beta x \tag{1}$$

by minimizing the L1-norm $\sum_{i=1}^{n} |y_i - \alpha - \beta x_i|$ where (x_i, y_i), $i = 1, 2, \ldots, n$, are n pairs of data values.

The early history of this criterion has been discussed recently by Plackett (1972). Although largely overshadowed by the Gauss/Legendre least-squares criterion, there has always been some interest in the L1-norm. Edgeworth (1887, 1888) pointed out that the L1-norm was appropriate when the law of error is double-exponential and he gave a method of computation. Rhodes (1930) gave a recursive method of computation for the "multiple regression" case and Singleton (1940) gave a geometric interpretation of the problem and an iterative method of computation which he proved to be convergent. Karst (1958) gave yet another method and also discussed the problem when (1) is constrained to pass through a given point, which may be taken to be the origin. More recently there has been an interest in linear programming algorithms. Wagner (1959) gave the first of these, Davis (1967) independently gave a

† F.A.O. Fellow, visiting from N. Poushkarov Institute of Soil Science, Sofia, Bulgaria.

similar method, whilst Barrodale and Young (1966) published the first computer algorithms as Algol procedures. Barrodale (1968) pointed out the robustness of L1-norm estimates to gross errors in the dependent variable, compared with least-squares estimates. Kiountouzis (1973) discussed simulation results and Spyropoulos *et al.* (1973) give an improved linear programming L1-norm algorithm in Algol. For the simple form (1) linear programming algorithms are cumbersome and therefore the algorithm used here is based on the Edgeworth (1888) method. The L1-norm fit must pass through at least two data points. Hence starting with a point P_1 the best point P_i to join it to is selected; P_i is then joined to its best companion P_j, etc. When P_i selects P_j and P_j selects P_i the L1-norm is minimized (Karst, 1958, p. 128).

The best point P_j to join to P_i is determined as follows. Without loss of generality we may assume P_i is at the origin so the L1-norm is

$$S = \sum_{k=1}^{n} |y_k - \beta x_k|.$$

S, considered as a function of β, therefore has slope

$$\sum_{k=1}^{n} |x_k| \operatorname{sign}(\beta - y_k/x_k). \tag{2}$$

Thus for all values of $\beta < \min(y_k/x_k)$, S decreases steeply, but the rate of decrease lessens as β successively passes increasing values of the slope y_k/x_k of the line $P_i P_k$. Eventually S starts to increase again and the least value of S occurs when j is taken to be the last value of k for which the slope of S is negative. This can be found by ranking in increasing order the values of y_k/x_k so that for any β the slope becomes

$$\sum_{y_k/x_k<\beta} |x_k| - \sum_{y_k/x_k>\beta} |x_k| = S_1 - S_2 \quad \text{(say)}.$$

But $S_1 + S_2 = \sum |x_k|$ a constant, so that the slope, which increases with β, may be written

$$2S_1 - \sum_{k=1}^{n} |x_k|.$$

Hence P_j is the point for which

$$S_1 = \sum_{k=1}^{j} |x_k| \leqslant \tfrac{1}{2} \sum_{k=1}^{n} |x_k| < \sum_{k=1}^{j+1} |x_k|.$$

With some pathological data (as, for example, four points at the corners of a square) the L1-norm solution is not unique. In tests, the algorithm has so far always found one of the possible solutions (which one depends on the order in which the data are presented), but there remains the possibility in cases of this kind that the iterative process described above might not always converge but merely cycle amongst several equal solutions. To protect the user against this contingency (which might be impossible) a test has been incorporated that gives a failure indication (see below); the fitted line is then taken to be the one joining the two last selected values of P_j.

STRUCTURE

SUBROUTINE LONESL (N, DX, DY, A, B, S, IT, DELTA, T, IFAULT)

Formal parameters

N	Integer	input: the number of points, n
DX	Real array (*N*)	input: the observed values (x_i), $i = 1, 2, \ldots, n$
DY	Real array (*N*)	input: the observed values (y_i), $i = 1, 2, \ldots, n$
A	Real	output: estimate of α
B	Real	output: estimate of β
S	Real	output: the sum of the moduli of the residuals
IT	Integer	output: number of iterative steps
DELTA	Real array (*N*)	output: the signed residuals
T	Real array (*N*)	: working array
IFAULT	Integer	output: see *Failure indications* below

Failure indications

IFAULT = 0 for a successful run
= 1 if a value of $n \leqslant 1$ is given
= 2 if convergence has not been achieved by the nth iteration (pathological data only)

Restrictions

The working array *T* is included with the formal parameters merely to allow variable length. Some users will prefer to declare *T* internally with a fixed length.

The local constant *PREC* is set to 10^{-6}, suitable for 4-byte (32 bit) single-precision real variables. This precision constant is used to check that no near-zero divisors occur in the statement with label 3. Big slopes are associated with near-zero divisors and these are set to 10^{19} by the local constant *BIG*.

TIME AND ACCURACY

Time depends on the number of steps of iteration which has an upper bound of n. In practice far fewer steps are needed and in a simulation experiment in which over 1000 lines were fitted, an average of three steps was needed for $n = 10$ and four steps for $n = 50$.

As the fitted line is determined by two points, the accuracy is the maximum real-number accuracy of the machine subject to the setting of *PREC* described above.

ACKNOWLEDGEMENT

This work was supported by a F.A.O. Fellowship.

REFERENCES

BARRODALE, I. (1968). L_1 approximation and the analysis of data. *Appl. Statist.*, **17**, 51–57.

BARRODALE, I. and YOUNG, A. (1966). Algorithms for best L_1 and L_∞ linear approximation on a discrete set. *Numer. Math.*, **8**, 295–306.

DAVIS, M. (1967). Linear approximation using the criterion of least total deviation. *J. R. Statist. Soc.* B, **29**, 101–109.

EDGEWORTH, F. Y. (1887). A new method of reducing observations relating to several quantities. *Phil. Mag.* (5th Ser.), **24**, 222–223.

—— (1888). On a new method of reducing observations relating to several quantities. *Phil. Mag.* (5th Ser.), **25**, 184–191.

KARST, O. J. (1958). Linear curve fitting using least deviations. *J. Amer. Statist. Ass.*, **53**, 118–132.

KIOUNTOUZIS, E. A. (1973). Linear programming techniques in regression analysis. *Appl. Statist.*, **22**, 69–73.

PLACKETT, R. (1972). Studies in the history of probability and statistics XXIX. The discovery of the method of least squares. *Biometrika*, **59**, 239–251.

RHODES, E. C. (1930). Reducing observations by the method of minimum deviations. *Phil. Mag.* (7th Ser.), **9**, 974–992.

SINGLETON, R. R. (1940). A method of minimizing the sum of the absolute values of deviations. *Ann. Math. Statist.*, **11**, 301–310.

SPYROPOULOS, K., KIOUNTOUZIS, E. and YOUNG, A. (1973). Discrete approximations in the L1-norm. *Computer J.*, **16**, 180–186.

WAGNER, H. M. (1959). Linear programming techniques for regression analysis. *J. Amer. Statist. Ass.*, **54**, 206–212.

```
      SUBROUTINE LONESL(N, DX, DY, A, B, S, IT, DELTA, T, IFAULT)
C
C        ALGORITHM AS 74  APPL. STATIST. (1974) VOL.23, NO.2
C
C        L1-NORM FIT OF A STRAIGHT LINE BY EDGEWORTH-KARST
C
      DIMENSION DX(N), DY(N), DELTA(N), T(N)
      INTEGER I, J, ALPHA, IP1, P, Q
      REAL PREC, SD, BIG, X, Y, DXA, DYA, T
      DATA PREC, BIG /1E-6, 1E19/
      IFAULT = 0
      IT = 0
      IF (N .LE. 1) GOTO 100
      IP1 = 0
      ALPHA = 1
C
C        CALCULATE SLOPES FROM CURRENT ORIGIN ALPHA
C
    1 IT = IT + 1
      IF (IT .GT. N) GOTO 10
      SD = 0.0
      DXA = DX(ALPHA)
      DYA = DY(ALPHA)
      DO 5 I = 1, N
      X = DX(I) - DXA
      Y = DY(I) - DYA
      IF (ABS(X) .GE. PREC) GOTO 3
      T(I) = BIG
      GOTO 4
    3 T(I) = Y / X
    4 DELTA(I) = FLOAT(I)
      SD = SD + ABS(X)
    5 CONTINUE
      SD = SD / 2.0
C
C        RANK SLOPES
C
      J = N - 1
    6 DO 7 I = 1, J
      X = DELTA(I)
      Y = DELTA(I + 1)
      P = INT(X + 0.5)
      Q = INT(Y + 0.5)
      IF (T(P) .LE. T(Q)) GOTO 7
      DELTA(I + 1) = X
      DELTA(I) = Y
    7 CONTINUE
      J = J - 1
      IF (J .GT. 0) GOTO 6
```

```
C        SELECT BEST LINE THROUGH ALPHA
C
      X = 0.0
    9 J = J + 1
      P = INT(DELTA(J) + 0.5)
      X = X + ABS(DX(P) - DXA)
      IF (X .LT. SD) GOTO 9
      IF (P .EQ. IP1) GOTO 11
      IP1 = ALPHA
      ALPHA = P
      GOTO 1
C
C        SET FINAL A, B, L1-NORM AND RESIDUALS
C
   10 IFAULT = 2
   11 B = T(P)
      A = DY(ALPHA) - B * DX(ALPHA)
      SD = 0.0
      DO 12 I = 1, N
      DELTA(I) = DY(I) - B * DX(I) - A
      SD = SD + ABS(DELTA(I))
   12 CONTINUE
      S = SD
      RETURN
  100 IFAULT = 1
      RETURN
      END
```

Remark AS R10

On Scale Selections for Scatter Diagrams

By G. J. ROWLANDS and RITA M. POCOCK

Agricultural Research Council Institute for Research on Animal Diseases, Compton, Berks

Given a set of data, Thayer and Storer (1969) provided a method (AS 21) for choosing, for a specified number of intervals, a reasonable scale for a graph which enclosed the maximum and minimum of the data. Both this algorithm and AS 44, an algorithm for scatter diagram plotting (Sparks, 1971), which incorporates the method in a simpler form, could be improved.

1. Neither takes due account of computer round-off error, and this can be particularly critical if the minimum value is a whole number. For instance, the minimum value B is first truncated to a suitable number of significant figures as follows (AS 44):

```
B = B * 10. ** KOUNT
IF (B .LT. 0.0 .AND. B .NE. AINT(B)) B = B - 1.0
B = AINT(B) / (10. ** KOUNT)
```

In a test on an IBM 360 computer with $B = 50$ and $KOUNT = -1$ the result of the first instruction was 4·99999 so that the final result was 40·0. Also the second statement, included so that a negative whole number is truncated correctly, does not necessarily work because, in view of the first statement, there is no guarantee that the computer recognizes B as a whole number. To prevent computer round-off errors of *whole* numbers, the statements may be replaced by: