KPG1_QNTLx

Finds a quantile in a (possibly weighted) set of data

Description:

The routine calculates the value of a specified quantile in a set of data values, which may be weighted. In concept (although not in practice) it sorts the supplied data values into ascending order along with their associated positive weights, if supplied. It then finds a quantile Q such that the sum of the weights associated with all data values less than Q is a specified fraction FRACT of the sum of all the weights supplied. If no weights are supplied, then each data value is assigned unit weight. There are two main applications of this algorithm:

a) To find specified quantiles of a distribution of data values for statistical purposes. In this case, the weights may optionally be used to represent the number of times each data value occurs. In such cases, it may be useful to regard the distribution as continuous, and therefore to interpolate linearly between data values when obtaining the result.

b) Alternatively, the values may represent residuals from some fitted function. In this case, by setting FRACT to 0.5, the "weighted median residual" may be found. This has the property that if it is subtracted from all the original residuals, then the weighted sum of the absolute values of the corrected residuals will be minimised. Thus, it may be used as the basis for iteratively finding an 'L1' fit. In such cases, the required result will be equal to one of the data values (or may lie mid-way between two of them) and interpolation between values is not normally required.

Invocation

CALL KPG1_QNTLx(USEWT, INTERP, FRACT, EL, X, W, IP, Q, STATUS)

Arguments

USEWT = LOGICAL (Given)

Whether or not the data have associated weights.

INTERP = LOGICAL (Given)

Whether or not interpolation between data values should be performed when obtaining the result.

FRACT = ? (Given)

The fraction specifying the required quantile, in the range 0.0 to 1.0.

EL = INTEGER (Given)

Number of data values.

X(*) = ? (Given)

Array of data values.

W(*) = ? (Given)

Array of associated positive weights (if required). This argument will only be referenced if USEWT is .TRUE..

IP(EL) = INTEGER (Given and Returned)

On entry, an array of pointers identifying which elements of X (and W if supplied) are to be considered. On exit, these pointers will have been permuted to access the specified data elements in an order which is more nearly sorted than before (although in general it will not represent a complete sort of the data).

Q = ? (Returned)

The value of the requested quantile.

STATUS = INTEGER (Given and Returned)

The global status.

Notes:

- There are versions of this routine for processing both REAL and DOUBLE PRECISION data; replace the "x" in the routine name by R or D as appropriate. The types of the FACT, X, W and Q arguments should match the routine being used.
- This routine is optimised for use when the number of data values is large. In general, only a partial sort of the data will be performed, so this routine will perform better than most other methods of finding quantiles, which typically require a complete sort.
- The order in which the input pointers are supplied in the array IP is arbitrary, but there will often be an efficiency advantage in supplying them so that they access the data in nearly-sorted order. Thus, re-supplying the array of pointers generated by a previous invocation of this routine (for the same or similar data) may be worthwhile.

Timing

Details of the asymptotic time required to execute the original SELECT algorithm are not altogether clear from the published papers. It appears that this algorithm may have better average performance than other methods and the time required may approximate to EL * LOG(MIN(K, EL - K + 1)) where K is the rank of the largest data value which is smaller than the quantile being sought. However, Sedgewick (see References) indicates that such algorithms should, in general, complete in time proportional to EL, so the above formula may be incorrect. When using weighted data, the time will be multiplied by a further factor reflecting the non-linearity of the cumulative weight versus rank function and the difficulty of inverting it.

References

- Comm. of the ACM, vol 18, no. 3 (March 1975), p165.
- Also see page 173.
- In addition, see the algorithm assessment by T. Brown in Collected Algorithms of the ACM, (algorithm no. 489).
- Sedgwick, R., 1988, "Algorithms" (Addison-Wesley).