LA-UR -81-1024

MASTER

CONF - 810342 - -1

TITLE: COMPUTING PERCENTILES OF LARGE DATA SETS

AUTHOR(S): Jo Ann Howell

SUBMITTED TO: Proceedings of Computer Science and Statistics 13th
Symposium on the Interface, March 12-13, 1981,
Pittsburgh, PA

University of California

By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes.

The Los Alamos Scientific Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy

LOS ALAMOS SCIENTIFIC LABORATORY

Post Office Box 1663 Los Alamos, New Mexico 87545

An Affirmative Action/Equal Opportunity Employer

COMPUTING PERCENTILES OF LARGE DATA SETS

Jo Ann Howell
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Abstract.

We describe an algorithm for finding percentiles of large data sets (those having 100,000 or more points). This algorithm does not involve sorting the entire data set. Instead, we sample the data and obtain a guess for the percentile. Then, using the guess we extract a subset of the original data through which we search for the true percentile.

A. Sampling for an Estimate

A popular method for computing percentiles in large data sets is to locate percentiles in a random sample from the data, although this method may not produce the true percentile. It is merely an estimate, whereas the method described here locates the Tth largest number in a set of N numbers, where N may be very large. More detail is given in Ref. 4.

We define a large data set as one with enough points that analysis by conventional methods is either very difficult or impossible. The number of points may range from thousands to millions or more. Although the algorithm works for a smaller number of points, when the number is less than about 5000, it is more efficient to sort the entire set. This cutoff depends on the memory size involved.

There are several parameters (Table I) involved in the algorithm that the user may vary to meet the particular needs of the problem. They are described in more detail in the text.

The first step in our algorithm is to predict the Tth largest number or the Pth percentile, where $P=100*(N-T)/N$. The prediction is made by taking a random sample of size KSAMP from the original N data points. The Pth percentile in this sample is used as an estimate to find the Pth percentile in the original data.

TABLE I
ALGORITHM PARAMETERS

Parameter	Description
N	An integer; the number of input data points.
T	An integer; we compute the Tth largest number from the large data set.
P	$P=100*(N-T)/N$; P is a percentile.
Y	An array of length KSAMP of data points that have been sampled from the large data set.
KSAMP	An integer; length of the array Y.
PCNTL	A number from 0.00 to 1.00; used to construct a window in the large data set.
PCNTH	A number from 0.00 to 1.00; used to construct a window in the large data set.
INDEXL	An integer; index into the array Y.
INDEXH	An integer; index into the array Y.
CUTOFL	$Y(INDEXL)$; defines the lower bound for the data to be extracted.
CUTOFH	$Y(INDEXH)$; defines the upper bound for the data to be extracted.
I	An integer; the number of data points larger than CUTOFH.
ICYR	An integer; the number of data points extracted.
KBUF	An integer; the size of the buffer area used to move the large data set in and out of memory (typically 5000 to 10,000).

In our program we call the array containing the sample Y. Assuming that Y is sorted from smallest to largest, we compute the indices $INDEXL = \max ((N -$

$T)/N * KBAMP - PCNTL * KBAMP, 1)$ and
 $INDEXH = \min((N - T)/N * KBAMP + PCNTH * KBAMP, KBAMP).$

These indices mark positions $(100 * PCNTL)\%$ below and $(100 * PCNTH)\%$ above the estimated percentile.

For example, if we select a sample of 1000 points from a large set of 100,000 data points, and we want to find the 50,000th largest number (the 50th percentile), then

$N = 100000$,
 $T = 50000$,
 $KBAMP = 1000$, and
 $P = 50$.

If we let $PCNTL = PCNTH = 0.05$, then $INDEXL$ and $INDEXH$ are respectively 450 and 550. The max and min are included to prevent the indices from pointing outside of the array Y , that is, to exclude nonpositive indices or indices larger than $KBAMP$.

Next, cutoff values are computed using the indices

$CUTOFL = Y(INDEXL)$

and

$CUTOFH = Y(INDEXH).$

These values are used as cutoffs on the original large data set to extract some data that is then searched for the true percentile.

In our example, the number $Y(500)$ is the estimate for the 50th percentile for the large data set. The upper and lower cutoffs are $Y(550)$ and $Y(450)$ which we use to extract data from the large data set.

If we want to be reasonably sure that the P th percentile for the large data set lies in the interval $(CUTOFL, CUTOFH)$, we must use some care in the selection of $PCNTL$ and $PCNTH$. These values can be chosen by constructing a distribution-free confidence interval for the P th percentile in the sample (Ref. 3). If PR is the probability that the percentile P of a set of size N lies in the interval $(CUTOFL, CUTOFH)$, then

$$PR = \sum_{i=INDEXL}^{INDEXH} \frac{N-i}{N} \frac{i}{P} \frac{N-i}{(1-P)}$$

Using the approximation,

$$PR = 1/\sqrt{2\pi} \int_a^b \frac{e^{-t^2/2}}{e} dt$$

where

$$a = (INDEXL - N * P) / \sqrt{N * P * (1 - P)}$$

and

$$b = (INDEXH - 1 - N * P) / \sqrt{N * P * (1 - P)}.$$

We construct a table of values (Table II) to use for $PCNTL$ and $PCNTH$ for the case $KBAMP=1000$.

TABLE II
SUGGESTED WINDOW VALUES

KBAMP	P	INDEXL	INDEXH	PR	PCNTL	PCNTH
1000	0.50	468	532	0.95	0.032	0.032
1000	0.75	720	775	0.95	0.030	0.025
1000	0.90	881	919	0.95	0.019	0.019
1000	0.95	936	964	0.95	0.014	0.014
1000	0.98	970	989	0.95	0.010	0.009
1000	0.99	984	998	0.96	0.006	0.008

B. Locating the True Percentile

From the original data, we extract all data points that are greater than or equal to $CUTOFL$ and also less than or equal to $CUTOFH$. We have considerably fewer data points in this extracted data set than in the original set. If the sample of size $KBAMP$ closely resembles the original large data set in distribution, then the extracted data will probably contain the true percentile. That is, the interval $(CUTOFL, CUTOFH)$ contains the P th percentile of the large data set. We can then use this smaller set to begin our search for the true percentile.

The number T is adjusted by the number of points in the original data set that were larger than $CUTOFH$. That is, if I is the number of points larger than $CUTOFH$, then we let $t = T - I$ and search for the t -th largest number in the extracted data set. For this search we use the Blum algorithm. (See Refs. 1, 2, and 3.)

C. Error Conditions

Several error conditions are signaled in the program by an error message. One of these is a missed true percentile in the extracted data. That is, $I > T$ or $I + ICTR < T$. At this point, the user can increase $KBAMP$ in the hope of getting a more representative sample or increase $PCNTL$ or $PCNTH$. It is easy to determine

by how much we missed the percentile and in which direction we missed it.

REFERENCES

- [1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974, pp. 97-99.
- [2] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, Time Bounds for Selection, J. Comput. and System Sci. 7 (1973), pp. 448-461.
- [3] H. A. David, Order Statistics, John Wiley, New York, 1970, pp. 13-15.
- [4] J. A. Howell, An Algorithm for Computing Percentiles of Large Data Sets, Los Alamos National Laboratory, report LA-UR 81-179.
- [5] D. E. Knuth, The Art of Computer Programming, vol. III, Addison-Wesley, Reading, MA, 1973, pp. 216-217.