

Simulation: Quantiles of Large Datasets Using Liu & Su Method

Rashidat Brobbey Shih-Wei Su

December 4, 2006

1 Background Information

Businesses all over the world are developing rapidly and as such information gathering and storage is increasingly becoming a challenge. Thousands of companies, such as insurance companies, banks, and manufacturing industries need to save huge private data. For example, customer's salary, average life of customers, or other personal qualities are some of the few important information(data) insurance companies may collect. However, as the population increases, these companies need large storage system to save their customers' data, and also need good statistical software to obtain useful information from analyzing these huge data. The development of the computer has been able to handle the first problem; that is how to save and store huge data. While we are satisfied by large storage of today's computer, unfortunately we face another issue. No business software at present can analyze more than one million data. For example, given 10 million dataset, it is a challenge if an insurance company wants to find the median of the life expectancy of human in order to determine the price of life insurance it should charge.

Solutions to this problem are now widely discussed. One interesting remedy in finding quantiles of large datasets is to use the frequency table method which was offered by Liu and Su in 2001, namely the L&S method. So the natural question is: What is L&S method? The main idea of L&S method is to use the frequency table to store the frequency of data instead of storing all values of the data. This is done to reduce the amount of memory space all the data may occupy. For example, if we use a frequency table which includes 2000 intervals and use this frequency table to store all frequency of one million data. It is obviously the memory will reduce from one million to 2000. So how

to do we get the Quantile of the data using information from the frequency table? Well, the L&S method suggests to create a four degree polynomial to model the distribution of data in each interval of the frequency table. So for example, if we want to create a four degree polynomial in the k th interval, there are five unknown parameters. So, we can use five frequencies which lie in $(k-2)$ th, $(k-1)$ th, k th, $(k+1)$ th, $(k+2)$ th interval to solve all five equations. That means L&S method tries to use the distribution of data among interval of $(k-2)$ th, $(k-1)$ th, k th, $(k+1)$ th, $(k+2)$ th to model the distribution of data in the k th interval.

2 Method of Simulation

We want to investigate and to see how good the L&S method is by simulation. In our simulation, we chose three factors of interest. Three factors are:

1. Distribution: Normal(0,1), Gamma(2,4), Exp(0.1), and Mixed Distribution
2. Sample Size: 500000 & 1000000
3. Amount of Intervals: 500 & 2000

In the mixed distribution case, we mixed the first three distributions to be our new distribution. In order to be consistent, we chose the same sample size. In other words, the sample size for each distribution in the mixed distribution case is of 166667 and 333333 respectively.

The method of evaluation we chose is the average difference between the estimated quantiles and the true quantiles of our data. Also, because generated 1000 independent datasets for the simulation, we get the average difference by adding all differences and dividing by 1000.

3 Tables of Results

Quantiles	No. of Intervals=500		No. of Intervals=2000	
	N=500000	N=1000000	N=500000	N=1000000
$\xi_{.25}$	-6.64	-20.10	5.40	-0.19
$\xi_{.5}$	13.99	0.50	5.11	3.49
$\xi_{.75}$	29.92	26.69	12.35	5.018

Table 1: Average Bias ($\times 10^{-6}$) of Quantiles Estimation for N(0,1)

Quantiles	No. of Intervals=500		No. of Intervals=2000	
	N=500000	N=1000000	N=500000	N=1000000
$\xi_{.25}$	-1.68	-0.17	1.16	0.98
$\xi_{.5}$	11.91	10.28	1.23	1.99
$\xi_{.75}$	16.51	19.10	5.07	3.98

Table 2: Average Bias($\times 10^{-6}$) of Quantiles Estimation for Gamma(2,4)

Quantiles	No. of Intervals=500		No. of Intervals=2000	
	N=500000	N=1000000	N=500000	N=1000000
$\xi_{.25}$	7.91	-79.84	1.26	7.78
$\xi_{.5}$	-15.95	-78.19	1.52	-0.48
$\xi_{.75}$	48.71	46.27	17.15	12.48

Table 3: Average Bias($\times 10^{-6}$) of Quantiles Estimation for Exp(0.1)

Quantiles	No. of Intervals=500		No. of Intervals=2000	
	N=500000	N=1000000	N=500000	N=1000000
$\xi_{.25}$	-4322.71	-4453.87	-291.85	-331.45
$\xi_{.5}$	2806.36	3046.92	467.14	566.64
$\xi_{.75}$	-239.09	-361.40	181.25	199.94

Table 4: Average Bias($\times 10^{-6}$) of Quantiles Estimation for Mixed Distribution of N(0,1), Gamma(2,4) and Exp(0.1)

4 Summary of Results

From the table of results, the larger the number of intervals we choose the better the estimated quantiles we get. That is when the number of intervals was 2000 the quantiles estimated was closer to the true quantiles than when the number of intervals was 500. This is because the range of the data is fixed and thus the larger number of intervals chosen will narrow the range of each interval of the frequency table. In L&S method, the author suggest that the bigger the sample size, the more accurate the simulated estimates will be. But in our simulation, the results we got for both sample size of 500000 and 1000000 are not significantly different. The results obtained in Table 4 for the mixed distribution looks bigger the other 3 distributions above. This may be due to the fact that the mixed distribution is more complicated than the other three

distribution, therefore the four degree polynomial is not good way to model this distribution, perhaps we need a higher degree polynomial. In conclusion, the results obtained from simulation seems reasonable. The average bias is significantly small because the values shown above is $\times 10^{-6}$. This shows that the L&S method is a good way of estimating quantiles of large data.

A R code for Simulation

Below is the code for the Normal Distribution. For the other distributions, we just do the substitutions accordingly.

```

q<-function(x){
dd1<-0
dd2<-0
dd3<-0
for (j in 1:1000){
x<-rexp(1000000,10)
N<-length(x)
A<-2000
quan<-matrix(0,A,6)
count<-rep(0,A)
constant<-matrix(0,A,5)
int<-(max(x)-min(x))/A
indix<-rep(0,A)

for(i in 1:(A+1)){
indix[i]<-min(x)+(i-1)*int
}

xx<-ceiling((x-min(x))/int)

for(i in 1:N){
count [xx [i]]<-count [xx [i]]+1
}
p1<-0
p2<-0
p3<-0
for(i in 3:(A-2)){
m1<-(indix[i]^4-(indix[i]+.25*int)^4)*(-.5*int)-(indix[i]^4-
```

```

(indix[i]+.5*int)^4)*(-.25*int)
n1<-(indix[i]^3-(indix[i]+.25*int)^3)*(-.5*int)-(indix[i]^3-
(indix[i]+.5*int)^3)*(-.25*int)
o1<-(indix[i]^2-(indix[i]+.25*int)^2)*(-.5*int)-(indix[i]^2-
(indix[i]+.5*int)^2)*(-.25*int)

m2<-(indix[i]^4-(indix[i]+.25*int)^4)*(-.75*int)-(indix[i]^4-
(indix[i]+.75*int)^4)*(-.25*int)
n2<-(indix[i]^3-(indix[i]+.25*int)^3)*(-.75*int)-(indix[i]^3-
(indix[i]+.75*int)^3)*(-.25*int)
o2<-(indix[i]^2-(indix[i]+.25*int)^2)*(-.75*int)-(indix[i]^2-
(indix[i]+.75*int)^2)*(-.25*int)

m3<-(indix[i]^4-(indix[i]+.25*int)^4)*(-int)-(indix[i]^4-
(indix[i+1])^4)*(-.25*int)
n3<-(indix[i]^3-(indix[i]+.25*int)^3)*(-int)-(indix[i]^3-
(indix[i+1])^3)*(-.25*int)
o3<-(indix[i]^2-(indix[i]+.25*int)^2)*(-int)-(indix[i]^2-
(indix[i+1])^2)*(-.25*int)

p1<-(-.5*int)*(count[i-2]-count[i-1])-(.25*int)*(count[i-2]
-count[i])
p2<-(.75*int)*(count[i-2]-count[i-1])-(.25*int)*(count[i-2]
-count[i+1])
p3<-(1*int)*(count[i-2]-count[i-1])-(.25*int)*(count[i-2]
-count[i+2])

constant[i,1]<-((p1*o2-p2*o1)*(n1*o3-n3*o1)-(p1*o3-p3*o1)*
(n1*o2-n2*o1))/((m1*o2-m2*o1)*(n1*o3-n3*o1)-(m1*o3-m3*o1)*
(n1*o2-n2*o1))
constant[i,2]<-(p1*o2-p2*o1-(m1*o2-m2*o1)*constant[i,1])/
(n1*o2-n2*o1)
constant[i,3]<-(p1-m1*constant[i,1]-n1*constant[i,2])/o1
constant[i,4]<-(count[i-2]-count[i-1]-(indix[i]^4-(indix[i]+
.25*int)^4)*constant[i,1]-(indix[i]^3-(indix[i]+.25*int)^3)*
constant[i,2]-(indix[i]^2-(indix[i]+.25*int)^2)*constant[i,3])/
(.25*int)
constant[i,5]<-count[i-2]-indix[i]^4*constant[i,1]-indix[i]^3*
constant[i,2]-indix[i]^2*constant[i,3]-indix[i]*constant[i,4]

```

```

k<-(count[i]-(constant[i,1]*(indx[i+1]^5-indix[i]^5)/5+
constant[i,2]*(indx[i+1]^4-indix[i]^4)/4+constant[i,3]*
(indix[i+1]^3-indix[i]^3)/3+constant[i,4]*(indx[i+1]^2-
indix[i]^2)/2+constant[i,5]*(indx[i+1]-indix[i])))/int
constant[i,5]<-constant[i,5]+k

```

```

quan[i,1]<-0
quan[i,2]<-(constant[i,1]*((indx[i]+.2*int)^5-indix[i]^5)/5+
constant[i,2]*((indx[i]+.2*int)^4-indix[i]^4)/4+constant[i,3]*
*((indx[i]+.2*int)^3-indix[i]^3)/3+constant[i,4]*((indx[i]+
.2*int)^2-indix[i]^2)/2+constant[i,5]*((indx[i]+.2*int)-indix[i]))
quan[i,3]<-(constant[i,1]*((indx[i]+.4*int)^5-indix[i]^5)/5+
constant[i,2]*((indx[i]+.4*int)^4-indix[i]^4)/4+constant[i,3]*
((indx[i]+.4*int)^3-indix[i]^3)/3+constant[i,4]*((indx[i]+
.4*int)^2-indix[i]^2)/2+constant[i,5]*((indx[i]+.4*int)-indix[i]))
quan[i,4]<-(constant[i,1]*((indx[i]+.6*int)^5-indix[i]^5)/5+
constant[i,2]*((indx[i]+.6*int)^4-indix[i]^4)/4+constant[i,3]*
((indx[i]+.6*int)^3-indix[i]^3)/3+constant[i,4]*((indx[i]+
.6*int)^2-indix[i]^2)/2+constant[i,5]*((indx[i]+.6*int)-indix[i]))
quan[i,5]<-(constant[i,1]*((indx[i]+.8*int)^5-indix[i]^5)/5+
constant[i,2]*((indx[i]+.8*int)^4-indix[i]^4)/4+constant[i,3]*
((indx[i]+.8*int)^3-indix[i]^3)/3+constant[i,4]*((indx[i]+
.8*int)^2-indix[i]^2)/2+constant[i,5]*((indx[i]+.8*int)-indix[i]))
quan[i,6]<-(constant[i,1]*((indx[i]+int)^5-indix[i]^5)/5+constant[i,2]*
((indx[i]+int)^4-indix[i]^4)/4+constant[i,3]*((indx[i]+int)^3-
indix[i]^3)/3+constant[i,4]*((indx[i]+int)^2-indix[i]^2)/2+
constant[i,5]*((indx[i]+int)-indix[i]))
}

```

```

q_1<-.25*N
q_2<-.5*N
q_3<-.75*N

```

```

z1<-0
z2<-0
z3<-0
k1<-1
k2<-1
k3<-1

```

```

while(k1<q_1){
z1<-z1+1
k1<-sum(count[1:z1])
}
while(k2<q_2){
z2<-z2+1
k2<-sum(count[1:z2])
}
while(k3<q_3){
z3<-z3+1
k3<-sum(count[1:z3])
}

h1<-(q_1-sum(count[1:(z1-1)]))/count[z1]
h2<-(q_2-sum(count[1:(z2-1)]))/count[z2]
h3<-(q_3-sum(count[1:(z3-1)]))/count[z3]
#print(c(h1,h2,h3))

h11<-ceiling(h1*5)
h22<-ceiling(h2*5)
h33<-ceiling(h3*5)
#print(c(h11,h22,h33))

r1<-indix[z1]+h11*int*.2-int*.2*(quan[z1,(h11+1)]-h1*count[z1])/
(quan[z1,(h11+1)]-quan[z1,h11])
r2<-indix[z2]+h22*int*.2-int*.2*(quan[z2,(h22+1)]-h2*count[z2])/
(quan[z2,(h22+1)]-quan[z2,h22])
r3<-indix[z3]+h33*int*.2-int*.2*(quan[z3,(h33+1)]-h3*count[z3])/
(quan[z3,(h33+1)]-quan[z3,h33])
#print(c(quan[z1,],h1*count[z1]))
#print(c(quan[z2,],h2*count[z2]))
#print(c(quan[z3,],h3*count[z3]))
d1<-(r1-quantile(x,.25))
d2<-(r2-quantile(x,.5))
d3<-(r3-quantile(x,.75))

dd1<-dd1+d1
dd2<-dd2+d2
dd3<-dd3+d3

```

```

#print(j)
#print(int)
}
dd1<-dd1/1000
dd2<-dd2/1000
dd3<-dd3/1000
}
print(c("the average bias of .25 quantile is",dd1))
print(c("the average bias of .5 quantile is",dd2))
print(c("the average bias of .75 quantile is",dd3))

}

```

References

- [1] Chao. M.T. and Lin, G.D(1993). The aysmptotic distributions of the re-medians. *Journal of Statistical Planning and Inference*, **37**, 1-11.
- [2] Knuth, Donald D.(1980). The Art of Computer Programming. Vol.3.(Addison-Wesley)
- [3] Rousseeuw, P.J and Bassett, G. W., Jr.(1990). The remedian: a rubust averaging method for larger data sets. *Journal of the American Statistical Association*, **85**, 97-104
- [4] Serfling, Robert J.(1984). Approximation Theorems of Mathematical Statistics, New York: Wiley
- [5] Liu and Su(2002) A Simple Computation Procedure of Basic Statistics For Mass Data