**Results from 15 RR files**

**DataShare SE Light Weight SE**

**------------ -------**

**0.1560 0**

**0.0785 0**

**0.0412 0**

**0.0888 0**

**0.0937 0**

**0.1060 0**

**0.0956 0**

**0.0988 0**

**0.0540 0**

**0.0284 0**

**0.0382 0**

**0.0815 0**

**0.0917 0**

**0.0299 0**

**0.1502 0**

**Note:**

* **Code under after %% are from authord in original cpp file**
* **In Matlab %% is used for comment therefore, c++ codes are commented and are replaced by matlab file**
* **%% ??? Q are the area’s of concern during code conversion**

% Sample\_entropy\_lightweight.cpp

% ------------------------------

%

%

% // g++ -Wall -O2 -shared -fPIC sample\_entropy\_lightweight.cpp -o sample\_entropy\_lightweight\_lib.so

%

% #include <stdlib.h>

% #include <math.h>

% typedef struct {

% double Dato;

% unsigned int p\_ini;

% } d\_o;

% int sort\_function( const void \*a, const void \*b );

% extern "C" double light(double \*x, int N, int m, double r)

% {

function[ans]= sampleEntropyLW(x,N,m,r)

%int Nm = N-m;

Nm = N-m;

%int i,j,k,ii,jj;

%int A=0,B=0;

%% ??? Q6 I think A and B are used as count not index..

A=0;

B=0;

d\_o = struct('Dato',{},'p\_ini',{});

%int r\_sup, i\_inf, i\_sup, i\_mez;

%double \*X;

% double D\_piu\_r;

% d\_o \*\_D\_ordinati;

D\_ordinati=d\_o();

% int \*originalPositions;

%double sum=0.0;

sum=0.0;

% for(i=0; i<N; i++)

% sum += x[i];

for i=1:N

sum=sum+x(i);

end

%double mean = sum/N;

%double standardDeviation=0.0;

mean = sum/N;

standardDeviation=0.0;

% for (i=0; i<N; i++)

% standardDeviation += (x[i]-mean)\*(x[i]-mean);

for i=1:N

standardDeviation = standardDeviation+ (x(i)-mean)\*(x(i)-mean);

end

standardDeviation = sqrt(standardDeviation/N);

r=r\*standardDeviation;

%X = (double \*)malloc(Nm\*sizeof(double));

% for (i=0;i<Nm;i++)

% {

% X[i]=x[i];

% for (j=1;j<m;j++)

% X[i]+=x[i+j];

% }

%

for i=1:Nm

X(i)=x(i);

for j=2:m %% ??? Q1 j index start from 1 in c means j should start from 2 in matlab

X(i)=X(i)+x(i+j);

end

end

%\_D\_ordinati = (d\_o \*)malloc(Nm\*sizeof(d\_o));

% for(i=0; i<Nm; i++)

% {

% \_D\_ordinati[i].Dato=X[i];

% \_D\_ordinati[i].p\_ini=i;

% }

for i=1:Nm

D\_ordinati(i).Dato=X(i);

D\_ordinati(i).p\_ini=i;

end

%qsort((void \*)\_D\_ordinati, (size\_t)Nm, sizeof(d\_o), sort\_function);

%https://uk.mathworks.com/matlabcentral/answers/397385-how-to-sort-a-structure-array-based-on-a-specific-field

%% ??? Q2 using sorting from matlab instead of qsort used in the c++ file...

[~,idx]=sort([D\_ordinati.Dato]);

D\_ordinati=D\_ordinati(idx);

%originalPositions = (int \*)malloc(Nm\*sizeof(int));

% for(i=0; i<Nm; i++)

% {

% X[i]=\_D\_ordinati[i].Dato;

% originalPositions[i]=\_D\_ordinati[i].p\_ini;

% }

for i=1:Nm

X(i)=D\_ordinati(i).Dato;

originalPositions(i)=D\_ordinati(i).p\_ini;

end

%free(\_D\_ordinati);

D\_ordinati={};

% for(i=0; i<Nm; i++)

% {

for i=1:Nm

% D\_piu\_r=X[i]+m\*r;

D\_piu\_r=X(i)+m\*r;

% if(D\_piu\_r >= X[Nm-1])

% r\_sup=Nm-1;

% else

% {

if D\_piu\_r >= X(Nm) %% ??? Q3 Nm-1 index in c is Nm im matlab

r\_sup=Nm-1;

else

% i\_inf=i;

% i\_sup=Nm-1;

i\_inf=i;

i\_sup=Nm; %% ??? Q4 not sure is Nm-1 is used as index or value

% while(i\_sup-i\_inf>1)

% {

% i\_mez=(i\_inf+i\_sup)>>1;

% if( X[i\_mez] > D\_piu\_r )

% i\_sup=i\_mez;

% else

% i\_inf=i\_mez;

% }

% r\_sup=i\_inf;

while i\_sup-i\_inf>1

%% i\_mez=(i\_inf+i\_sup)>>1???

%%??? Q5 I am using bitshift from matlab ..I have check with c++ compiler both gives same result

i\_mez = bitshift(i\_inf+i\_sup, -1);

if X(i\_mez) > D\_piu\_r

i\_sup=i\_mez;

else

i\_inf=i\_mez;

end

end

r\_sup=i\_inf;

% }

end

% ii=originalPositions[i];

ii=originalPositions(i);

% for(j=i+1; j<=r\_sup; j++)

% {

% jj=originalPositions[j];

for j=i+1:j<=r\_sup

% for (k=0;k<m;k++)

% if (fabs(x[ii+k]-x[jj+k])>r)

% break;

% if (k==m)

% {

% B++;

% if (fabs(x[ii+m]-x[jj+m])<=r)

% A++;

% }

for k=1:m

if abs(x(ii+k)-x(ii+k))>r

break

end

end

if k==m

B=B+1;

if abs(x(ii+k)-x(ii+k))<=r

A=A+1;

end

end

% }

end

% }

end

% if (A\*B==0)

% return log((N-m)\*(N-m-1));

% else

% return -log(1.0\*A/B);

if A\*B ==0

%% ??? Q7 hope this is natural log....

ans= log((N-m)\*(N-m-1));

return;

else

ans= -log(1.0\*A/B);

return;

end

%}

end

% int sort\_function( const void \*a, const void \*b)

% {

% return ( ((d\_o \*)a)->Dato > ((d\_o \*)b)->Dato ) ? 1 : -1;

% }