**RecSys vsu’s source code**

By Vladimir Nikulin

*Vyatka State University, Russia*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Copyright [2013] [vsu Team]**

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.

You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS"

BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,

either express or implied. See the License for the specific language governing permissions and limitations under the License.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**SOURCE CODE:**

library(rjson)

# Convert raw JSON file into managable data frame

UnpackJSON <- function(filePath){

con <- file(filePath, "r")

input <- readLines(con)

jsonData <- sapply(input,fromJSON)

close(con)

df <- data.frame(jsonData)

temp <- rownames(df)

df <- as.data.frame(t(df))

colnames(df) <- temp

rownames(df) <- NULL

return(df)

}

# Convert the nested lists into regular vectors

UnlistJSON <- function(df){

for(i in 1:ncol(df)) {

temp <- unlist(df[,i])

names(temp) <- NULL

df[,i] <- temp

}

return(df)

}

# Training User data

filePath <- "c:/contest/Kaggle/RecSys/data/train\_review\_N3.txt"

user <- UnpackJSON(filePath)

user <- user[,c('user\_id', 'business\_id', 'stars')]

user <- UnlistJSON(user)

write.table(user, file = "c:/contest/Kaggle/RecSys/data1/trn\_review\_N3.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, sep = " ")

########################################################

library(rjson)

# Convert raw JSON file into managable data frame

UnpackJSON <- function(filePath){

con <- file(filePath, "r")

input <- readLines(con)

jsonData <- sapply(input,fromJSON)

close(con)

df <- data.frame(jsonData)

temp <- rownames(df)

df <- as.data.frame(t(df))

colnames(df) <- temp

rownames(df) <- NULL

return(df)

}

# Convert the nested lists into regular vectors

UnlistJSON <- function(df){

for(i in 1:ncol(df)) {

temp <- unlist(df[,i])

names(temp) <- NULL

df[,i] <- temp

}

return(df)

}

# Training User data

filePath <- "c:/contest/Kaggle/RecSys/yelp\_test\_set/yelp\_test\_set\_review.json"

user <- UnpackJSON(filePath)

user <- user[,c('user\_id', 'business\_id')]

user <- UnlistJSON(user)

write.table(user, file = "c:/contest/Kaggle/RecSys/data1/tst\_review.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, sep = " ")

########################################################

#!/usr/local/bin/perl

#\*\*\*\* 16th May 2013 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### transform\_review.pl ####################

#===========================================

open(f1, "c:/contest/Kaggle/RecSys/data1/trn\_review\_N1.txt");

open(f2, "c:/contest/Kaggle/RecSys/data1/trn\_review\_N2.txt");

open(f3, "c:/contest/Kaggle/RecSys/data1/trn\_review\_N3.txt");

open(ft, "c:/contest/Kaggle/RecSys/data1/tst\_review.txt");

#mkdir("c:/contest/Kaggle/RecSys/data1/");

open(fa, ">c:/contest/Kaggle/RecSys/data1/library\_user.txt");

open(fb, ">c:/contest/Kaggle/RecSys/data1/library\_business.txt");

open(ff, ">c:/contest/Kaggle/RecSys/data1/trn\_review\_num.txt");

open(fc, ">c:/contest/Kaggle/RecSys/data1/tst\_review\_num.txt");

open(fx, ">c:/contest/Kaggle/RecSys/data1/trace\_review\_num.txt");

#===========================================

for($i=0;$i<=1;$i++){$num{$i} = 0;}

#===========================================

while(defined($\_ = <f1>)){

chomp $\_;

@fields = split(/ /,$\_);

for($i=0;$i<=1;$i++){

$a = @fields[$i];

if(defined($s{$i}{$a})){

$z{$i}{$a} = $z{$i}{$a} + 1;

}

else{

$num{$i} = $num{$i} + 1;

$ii = $num{$i};

$s{$i}{$a} = $ii;

$z{$i}{$a} = 1;

$q{$i}{$ii} = $a;

}

}

}

print "$num{0} $num{1}\n";

print fx "$num{0} $num{1}\n";

#===========================================

while(defined($\_ = <f2>)){

chomp $\_;

@fields = split(/ /,$\_);

for($i=0;$i<=1;$i++){

$a = @fields[$i];

if(defined($s{$i}{$a})){

$z{$i}{$a} = $z{$i}{$a} + 1;

}

else{

$num{$i} = $num{$i} + 1;

$ii = $num{$i};

$s{$i}{$a} = $ii;

$z{$i}{$a} = 1;

$q{$i}{$ii} = $a;

}

}

}

print "$num{0} $num{1}\n";

print fx "$num{0} $num{1}\n";

#===========================================

while(defined($\_ = <f3>)){

chomp $\_;

@fields = split(/ /,$\_);

for($i=0;$i<=1;$i++){

$a = @fields[$i];

if(defined($s{$i}{$a})){

$z{$i}{$a} = $z{$i}{$a} + 1;

}

else{

$num{$i} = $num{$i} + 1;

$ii = $num{$i};

$s{$i}{$a} = $ii;

$z{$i}{$a} = 1;

$q{$i}{$ii} = $a;

}

}

}

print "$num{0} $num{1}\n";

print fx "$num{0} $num{1}\n";

#===========================================

while(defined($\_ = <ft>)){

chomp $\_;

@fields = split(/ /,$\_);

for($i=0;$i<=1;$i++){

$a = @fields[$i];

if(defined($s{$i}{$a})){

$z{$i}{$a} = $z{$i}{$a} + 1;

}

else{

$num{$i} = $num{$i} + 1;

$ii = $num{$i};

$s{$i}{$a} = $ii;

$z{$i}{$a} = 1;

$q{$i}{$ii} = $a;

}

}

}

print "$num{0} $num{1}\n";

print fx "$num{0} $num{1}\n";

#===========================================

$k = $num{0};

for($i=1;$i<=$k;$i++){

$a = $q{0}{$i};

$b = $s{0}{$a};

$m = $z{0}{$a};

print fa "$a $b $m\n";

}

#===========================================

$k = $num{1};

for($i=1;$i<=$k;$i++){

$a = $q{1}{$i};

$b = $s{1}{$a};

$m = $z{1}{$a};

print fb "$a $b $m\n";

}

#===========================================

close(ft);

close(f1);

close(f2);

close(f3);

#===========================================

open(f1, "c:/contest/Kaggle/RecSys/data1/trn\_review\_N1.txt");

open(f2, "c:/contest/Kaggle/RecSys/data1/trn\_review\_N2.txt");

open(f3, "c:/contest/Kaggle/RecSys/data1/trn\_review\_N3.txt");

open(ft, "c:/contest/Kaggle/RecSys/data1/tst\_review.txt");

#===========================================

for($i=0;$i<=1;$i++){$num{$i} = 0;}

#===========================================

while(defined($\_ = <f1>)){

@fields = split(/ /,$\_);

for($i=0;$i<=1;$i++){

$a = @fields[$i];

if(defined($s{$i}{$a})){

$b = $s{$i}{$a};

print ff "$b ";

}

else{

$num{$i} = $num{$i} + 1;

print ff "0 ";

}

}

$a = @fields[$i];

print ff "$a";

}

print "$num{0} $num{1}\n";

print fx "$num{0} $num{1}\n";

#===========================================

while(defined($\_ = <f2>)){

@fields = split(/ /,$\_);

for($i=0;$i<=1;$i++){

$a = @fields[$i];

if(defined($s{$i}{$a})){

$b = $s{$i}{$a};

print ff "$b ";

}

else{

$num{$i} = $num{$i} + 1;

print ff "0 ";

}

}

$a = @fields[$i];

print ff "$a";

}

print "$num{0} $num{1}\n";

print fx "$num{0} $num{1}\n";

#===========================================

while(defined($\_ = <f3>)){

@fields = split(/ /,$\_);

for($i=0;$i<=1;$i++){

$a = @fields[$i];

if(defined($s{$i}{$a})){

$b = $s{$i}{$a};

print ff "$b ";

}

else{

$num{$i} = $num{$i} + 1;

print ff "0 ";

}

}

$a = @fields[$i];

print ff "$a";

}

print "$num{0} $num{1}\n";

print fx "$num{0} $num{1}\n";

#===========================================

while(defined($\_ = <ft>)){

chomp $\_;

@fields = split(/ /,$\_);

for($i=0;$i<=1;$i++){

$a = @fields[$i];

if(defined($s{$i}{$a})){

$b = $s{$i}{$a};

print fc "$b ";

}

else{

$num{$i} = $num{$i} + 1;

print fc "0 ";

}

}

print fc "\n";

}

print "$num{0} $num{1}\n";

print fx "$num{0} $num{1}\n";

#===========================================

close(ff);

close(fa);

close(fb);

close(fc);

close(ft);

close(f1);

close(f2);

close(f3);

close(fx);

######################################################

//\*\*\*\*\*\*\*\*\*\*\*\* 17th April 2013 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\* gmf\_RecSys\_CV.c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\* RecSys 2013 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//###########################################################

#include<math.h>

#include<stdlib.h>

#include<stdio.h>

main()

{

int i,i1,i2,i3,ii,ii2,j,jj,jj2,k1,k2,n1,n2;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

const int nr=229907; //\*\*\* training

const int nt=22956; //\*\*\* testing

const int nu=51296;

const int ni=12742;

const int delta=20;

const int m=30;

const int k=3;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

const float H=32767.0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

float gamma,Q,error,S,S1,S2,Z,Z0,qnr,p1,p2,q,q1,q2;

//================================================

int \*A,\*B,\*index,\*T,\*u,\*v,\*f,\*g;

float \*fact1,\*fact2,\*solut\_tst;

//================================================

FILE \*ff,\*fa,\*fi,\*f1,\*f2;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ff=fopen("c:/contest/Kaggle/RecSys/data1/trn\_review\_num.txt","r");

fa=fopen("c:/contest/Kaggle/RecSys/data1/tst\_review\_num.txt","r");

fi=fopen("c:/contest/Kaggle/RecSys/param/index\_CV22956.txt","r");

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

f1=fopen("c:/contest/Kaggle/RecSys/gmf/trace\_gmf10.txt","w");

f2=fopen("c:/contest/Kaggle/RecSys/gmf/tst\_gmf10.txt","w");

//++++++++++++++++++++++++++++++++++++++++++++++++

A = (int \*)calloc(1000001, sizeof(int));

B = (int \*)calloc(300001, sizeof(int));

index = (int \*)calloc(230001, sizeof(int));

T = (int \*)calloc(100001, sizeof(int));

u = (int \*)calloc(200001, sizeof(int));

v = (int \*)calloc(60001, sizeof(int));

f = (int \*)calloc(120001, sizeof(int));

g = (int \*)calloc(25001, sizeof(int));

fact1 = (float \*)calloc(5000001, sizeof(int));

fact2 = (float \*)calloc(900001, sizeof(int));

solut\_tst = (float \*)calloc(23001, sizeof(int));

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if(A==NULL | T==NULL){

printf( "Can't allocate memory\n" );

scanf("%d",&i);

free(A);

free(T);

return 0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if(B==NULL | index==NULL){

printf( "Can't allocate memory\n" );

scanf("%d",&i);

free(B);

free(index);

return 0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if(u==NULL | v==NULL){

printf( "Can't allocate memory\n" );

scanf("%d",&i);

free(u);

free(v);

return 0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if(f==NULL | g==NULL){

printf( "Can't allocate memory\n" );

scanf("%d",&i);

free(f);

free(g);

return 0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if(fact1==NULL | fact2==NULL){

printf( "Can't allocate memory\n" );

scanf("%d",&i);

free(fact1);

free(fact2);

return 0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if(solut\_tst==NULL){

printf( "Can't allocate memory\n" );

scanf("%d",&i);

free(solut\_tst);

return 0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nt;i++){

solut\_tst[i]=0.0;

for(j=1;j<=5;j++){

fscanf(fi,"%d",&i1);

B[i+nt\*(j-1)]=i1;

}

fscanf(fi,"\n");

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nr;i++){

fscanf(ff,"%d",&i1);

A[i]=i1;

fscanf(ff,"%d",&i2);

A[i+nr]=i2;

fscanf(ff,"%d\n",&i3);

A[i+2\*nr]=i3;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for(i=1;i<=nt;i++){

fscanf(fa,"%d",&i1);

T[i]=i1;

fscanf(fa,"%d\n",&i2);

T[i+nt]=i2;

}

for(i=1;i<=nr;i++)index[i]=1;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

S2=0.0;

for(jj2=1;jj2<=5;jj2++){

//###########################################################

for(i=1;i<=nt;i++){

i1=B[i+nt\*(jj2-1)];

index[i1]=0;

}

//###########################################################

for(i=1;i<=nu;i++){

u[i]=0;

f[i]=0;

}

for(i=1;i<=ni;i++){

v[i]=0;

g[i]=0;

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

S=0.0;

for(i=1;i<=nr;i++){

if(index[i]==1){

i1=A[i];

u[i1]++;

i2=A[i+nr];

v[i2]++;

i3=A[i+2\*nr];

f[i1]+=i3;

g[i2]+=i3;

S+=i3;

}

}

qnr=nr-nt;

S1=S/qnr;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

i1=0;

for(i=1;i<=nu;i++){

if(u[i]>=delta){

i1++;

u[i+nu]=i1;

u[i1+2\*nu]=i;

}

}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

i2=0;

for(i=1;i<=ni;i++){

if(v[i]>=delta){

i2++;

v[i+ni]=i2;

v[i2+2\*ni]=i;

}

}

printf("%d %d %d\n",jj2,i1,i2);

n1=i1;

n2=i2;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

gamma=0.01;

Z0=99999999.9;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for(i=1;i<=n1;i++){

for(j=1;j<=k;j++)fact1[i+n1\*(j-1)]=0.01\*(rand()/H-0.5);

}

//=============================================

for(i=1;i<=n2;i++){

for(j=1;j<=k;j++)fact2[i+n2\*(j-1)]=0.01\*(rand()/H-0.5);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main:

for(jj=1;jj<=m;jj++){

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Z=0.0;

ii=0;

for(i=1;i<=nr;i++){

if(index[i]==1){

i1=A[i];

i2=A[i+nr];

if(u[i1+nu]>=1 && v[i2+ni]>=1){

k1=u[i1+nu];

k2=v[i2+ni];

S=0.0;

for(j=1;j<=k;j++)S+=fact1[k1+n1\*(j-1)]\*fact2[k2+n2\*(j-1)];

error=A[i+2\*nr]-S;

for(j=1;j<=k;j++){

Q=fact1[k1+n1\*(j-1)]\*fact2[k2+n2\*(j-1)];

fact1[k1+n1\*(j-1)]+=gamma\*error\*fact2[k2+n2\*(j-1)];

error+=Q-fact1[k1+n1\*(j-1)]\*fact2[k2+n2\*(j-1)];

Q=fact1[k1+n1\*(j-1)]\*fact2[k2+n2\*(j-1)];

fact2[k2+n2\*(j-1)]+=gamma\*error\*fact1[k1+n1\*(j-1)];

error+=Q-fact1[k1+n1\*(j-1)]\*fact2[k2+n2\*(j-1)];

}

ii++;

Z+=error\*error;

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

qnr=ii;

Z=sqrt(Z/qnr);

if(Z0>Z)Z0=Z;

printf("%d %d %9.8f %9.8f %9.8f %d %d %d\n",jj2,jj,Z,Z0,S2,ii,n1,n2);

fprintf(f1,"%d %d %9.8f %9.8f %9.8f %d %d %d\n",jj2,jj,Z,Z0,S2,ii,n1,n2);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

S2=0.0;

ii2=0;

for(i=1;i<=nr;i++){

if(index[i]==0){

i1=A[i];

i2=A[i+nr];

ii2++;

if(u[i1+nu]>=1 && v[i2+ni]>=1){

k1=u[i1+nu];

k2=v[i2+ni];

S=0.0;

for(j=1;j<=k;j++)S+=fact1[k1+n1\*(j-1)]\*fact2[k2+n2\*(j-1)];

if(S<1.0)S=1.0;

if(S>5.0)S=5.0;

S2+=(A[i+2\*nr]-S)\*(A[i+2\*nr]-S);

}

else{

if(u[i1]>=1 && v[i2]>=1){

q1=u[i1];

q2=v[i2];

p1=q1/(q1+q2);

p2=q2/(q1+q2);

S=p1\*f[i1]/q1+p2\*g[i2]/q2;

}

else if(u[i1]>=1){

q1=u[i1];

S=f[i1]/q1;

}

else if(v[i2]>=1){

q2=v[i2];

S=g[i2]/q2;

}

else S=S1;

if(S<1.0)S=1.0;

if(S>5.0)S=5.0;

S2+=(A[i+2\*nr]-S)\*(A[i+2\*nr]-S);

}

}

}

qnr=ii2;

S2=sqrt(S2/qnr);

//###########################################################

for(i=1;i<=nt;i++){

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

i1=B[i+nt\*(jj2-1)];

index[i1]=1;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

i1=T[i];

i2=T[i+nt];

if(u[i1+nu]>=1 && v[i2+ni]>=1){

k1=u[i1+nu];

k2=v[i2+ni];

S=0.0;

for(j=1;j<=k;j++)S+=fact1[k1+n1\*(j-1)]\*fact2[k2+n2\*(j-1)];

if(S<1.0)S=1.0;

if(S>5.0)S=5.0;

solut\_tst[i]+=S;

}

else{

if(u[i1]>=1 && v[i2]>=1){

q1=u[i1];

q2=v[i2];

p1=q1/(q1+q2);

p2=q2/(q1+q2);

S=p1\*f[i1]/q1+p2\*g[i2]/q2;

}

else if(u[i1]>=1){

q1=u[i1];

S=f[i1]/q1;

}

else if(v[i2]>=1){

q2=v[i2];

S=g[i2]/q2;

}

else S=S1;

if(S<1.0)S=1.0;

if(S>5.0)S=5.0;

solut\_tst[i]+=S;

}

}

//###########################################################

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for(i=1;i<=nt;i++){

q=solut\_tst[i]/5.0;

fprintf(f2,"%6.4f\n",q);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fclose(ff);

fclose(fa);

fclose(fi);

fclose(f1);

fclose(f2);

return(0);

}