% day/night specific genes

AT\_circadian\_day\_spec=cell(50000,2);

AT\_circadian\_night\_spec=cell(50000,2);

c=1;

d=1;

SIZE1=size(AT\_circadian\_day\_coexp);

for i=1:SIZE1(1)

i

for j=1:SIZE1(1)

if i>j

if AT\_circadian\_day\_coexp(i,j)>=0.8 & (AT\_circadian\_night\_coexp(i,j)>=-0.2 & AT\_circadian\_night\_coexp(i,j)<=0.2) AT\_circadian\_day\_spec(c,1)=AT\_circadian\_genes(i,2); AT\_circadian\_day\_spec(c,2)=AT\_circadian\_genes(j,2);

c=c+1;

elseif AT\_circadian\_night\_coexp(i,j)>=0.8 & (AT\_circadian\_day\_coexp(i,j)>=-0.2 & AT\_circadian\_day\_coexp(i,j)<=0.2)

AT\_circadian\_night\_spec(d,1)=AT\_circadian\_genes(i,2);

AT\_circadian\_night\_spec(d,2)=AT\_circadian\_genes(j,2);

d=d+1;

end

end

end

end

% roc curve

[xroc, yroc, Area] = roc2(x,alpha)

hold on

HR1=plot(xroc,yroc,'r.-');

hold on

HRC1=plot([0 1],[0 1],'k');

hold off

xlabel('False positive rate (1-Specificity)')

ylabel('True positive rate (Sensitivity)')

title('ROC curve')

axis square

% removedRF

COMPLETE\_VOTES1=zeros(134,2,1000);

COMPLETE\_Y\_hat11=zeros(134,1,1000);

REMOVED\_VOTES1=zeros(134,2,1000,210);

REMOVED\_Y\_hat11=zeros(134,1,1000,210);

REMOVED1\_VOTES1=zeros(134,2,1000,21);

REMOVED1\_Y\_hat11=zeros(134,1,1000,21);

WHICH\_CLASS=1;

for i=WHICH\_CLASS:WHICH\_CLASS

for j=1:1000

CAT=ELEMENT\_all\_together\_cat==i;

CAT2(CAT==1)=1;

CAT2(CAT==0)=2;

DANE1=SANDRA\_SIMULINK\_ELEMENT(CAT2==1,:);

DANE2=SANDRA\_SIMULINK\_ELEMENT(CAT2==2,:);

SAJZ1=size(DANE1);

RANDY=randperm(SAJZ1(1));

DANE3=[DANE1;DANE2(RANDY(1:SAJZ1(1)),:)];

CAT3=2\*ones(SAJZ1(1)\*2,1);

CAT3(1:SAJZ1(1))=1;

[A B C D]=RF\_random\_set(DANE3,CAT3,2);

COMPLETE=classRF\_train(A,B,1000,sqrt(21));

[Y\_hat1 votes] = classRF\_predict(C,COMPLETE);

eval(['COMPLETE\_C' num2str(i) '\_' num2str(j) '=C;'])

eval(['COMPLETE\_D' num2str(i) '\_' num2str(j) '=D;'])

eval(['COMPLETE\_VOTES' num2str(i) '(:,1:2,' num2str(j) ')=votes;'])

eval(['COMPLETE\_Y\_hat1' num2str(i) '(:,1,' num2str(j) ')=Y\_hat1;'])

c=1;

for k=1:21

for l=1:21

if k>l

i

j

k

SAJZ3=size(C);

RAND1=randperm(SAJZ3(1));

RAND2=randperm(SAJZ3(1));

REMOVED\_C=C;

REMOVED\_C(:,k)=C(RAND1,k);

REMOVED\_C(:,l)=C(RAND2,l);

[Y\_hat1 votes] = classRF\_predict(REMOVED\_C,COMPLETE);

eval(['REMOVED\_VOTES' num2str(i) '(:,1:2,' num2str(j) ',c)=votes;']);

eval(['REMOVED\_Y\_hat1' num2str(i) '(:,1,' num2str(j) ',c)=Y\_hat1;']);

c=c+1;

end

end

end

for m=1:21

SAJZ3=size(C);

RAND1=randperm(SAJZ3(1));

REMOVED\_C=C;

REMOVED\_C(:,k)=C(RAND1,m);

[Y\_hat1 votes] = classRF\_predict(REMOVED\_C,COMPLETE);

eval(['REMOVED1\_VOTES' num2str(i) '(:,1:2,' num2str(j) ',m)=votes;']);

eval(['REMOVED1\_Y\_hat1' num2str(i) '(:,1,' num2str(j) ',m)=Y\_hat1;']);

c=c+1;

end

end

end

%Timelag

[ output ] = time\_delay( input\_matrix, genes, TF\_list, corrcut )

% 1) input\_matrix - rows are genes, columns are time points

% 2) TF\_list - vector size of number of genes, 1/0 1 indicating Transcription

% factor, 0 any other gene

% 3) corrcut - correlation cutoff

% 4) genes - list of gene names, same order as input matrix

% delay set 1 time point in this case 4h

output=cell(500,3);

c=1;

SIZE1=size(input\_matrix);

VECTOR1=zeros(1,SIZE1(2));

for i=1:SIZE1(1)

if TF\_list(i)>0

for j=1:SIZE1(1)

if i~=j

VECTOR1(2:end)=input\_matrix(i,1:end-1);

VECTOR1(1)=input\_matrix(i,end);

CORRELATION=corr(VECTOR1',input\_matrix(j,:)');

if CORRELATION>=corrcut

output(c,1)=genes(i);

output(c,2)=genes(j);

output(c,3)={CORRELATION};

c=c+1;

end

end

end

end

end

%consurf

function [ consurf\_sub\_nr consurf\_final ] = consurf( seq )

% Each row in 'seq' is one Amino Acid sequence

params={'PDB\_yes\_no','no','MSA\_yes\_no','yes','msa\_FILE',d,'user\_email','sandra@cs.rhul.ac.uk','msa\_SEQNAME','nazwa','TREE\_yes\_no','no','DNA\_AA','Nuc','submit','submit'};

[n m]=size(seq);

consurf\_sub\_nr=cell(n,1);

progressbar('ConSurf')

SIZE1=size(seq);

for i=1:SIZE1(1)

aaa=ortologi\_map(i,:)>0;

ss1=sum(aaa);

hedy=cell(ss1,1);

seqy=cell(ss1,1);

ccc=1;

for zz=1:12

if aaa(zz)>0

switch zz

case 1

hedy(ccc)=s\_header\_oryza(ortologi\_map(i,1));

seqy(ccc)=s\_seq\_oryza\_strand(ortologi\_map(i,1));

ccc=ccc+1;

case 2

hedy(ccc)=s\_header\_lyrata(ortologi\_map(i,2));

seqy(ccc)=s\_seq\_lyrata\_strand(ortologi\_map(i,2));

ccc=ccc+1;

case 3

hedy(ccc)=s\_header\_brachy(ortologi\_map(i,3));

seqy(ccc)=s\_seq\_brachy\_strand(ortologi\_map(i,3));

ccc=ccc+1;

case 4

hedy(ccc)=s\_header\_papaya(ortologi\_map(i,4));

seqy(ccc)=s\_seq\_papaya\_strand(ortologi\_map(i,4));

ccc=ccc+1;

case 5

hedy(ccc)=s\_header\_glycine(ortologi\_map(i,5));

seqy(ccc)=s\_seq\_glycine\_strand(ortologi\_map(i,5));

ccc=ccc+1;

case 6

hedy(ccc)=s\_header\_manihot(ortologi\_map(i,6));

seqy(ccc)=s\_seq\_manihot\_strand(ortologi\_map(i,6));

ccc=ccc+1;

case 7

hedy(ccc)=s\_header\_medicago(ortologi\_map(i,7));

seqy(ccc)=s\_seq\_medicago\_strand(ortologi\_map(i,7));

ccc=ccc+1;

case 8

hedy(ccc)=s\_header\_populus(ortologi\_map(i,8));

seqy(ccc)=s\_seq\_populus\_strand(ortologi\_map(i,8));

ccc=ccc+1;

case 9

hedy(ccc)=s\_header\_ricinus(ortologi\_map(i,9));

seqy(ccc)=s\_seq\_ricinus\_strand(ortologi\_map(i,9));

ccc=ccc+1;

case 10

hedy(ccc)=s\_header\_sorghum(ortologi\_map(i,10));

seqy(ccc)=s\_seq\_sorghum\_strand(ortologi\_map(i,10));

ccc=ccc+1;

case 11

hedy(ccc)=s\_header\_zea(ortologi\_map(i,11));

seqy(ccc)=s\_seq\_zea\_strand(ortologi\_map(i,11));

ccc=ccc+1;

case 12

hedy(ccc)=s\_header\_thaliana(ortologi\_map(i,12));

seqy(ccc)=s\_seq\_thaliana\_strand(ortologi\_map(i,12));

ccc=ccc+1;

end

end

end

for zz=1:ss1

h1(zz,1).Header=hedy{zz,1};

h1(zz,1).Sequence=seqy{zz,1};

end

if ss1>4

ma=multialign(h1);

clear h1

multialignwrite('d:\alajn.aln',ma,'header','CLUSTAL 2.0.12 multiple sequence alignment');

f=fopen('d:\alajn.aln');

d=fread(f,Inf,'\*uint8');

fclose(f);

params(10)=ortologi(i,1);

consurf\_out=urlreadpost('http://consurf.tau.ac.il/cgi-bin/new\_consurf\_with\_DNA.cgi',params);

str1=strfind(consurf\_out,'<title>ConSurf run no. ');

str2=strfind(consurf\_out,' MSA File: dummy</title>');

str3=consurf\_out(str1(1)+23:str2(1)-1);

consurf\_sub\_nr(i)={str3};

pause(5);

end

end

url1='http://consurf.tau.ac.il/results/';

url2='/consurf.grades';

progressbar('ConSurf Final')

consurf\_output=cell(n,1);

for i=1:n

progressbar(i/n)

aaa=urlread([url1 char(consurf\_sub\_nr(i)) '/output.php']);

aaa2=strfind(aaa,'FINISHED');

if aaa2>0

cons=urlread([url1 char(consurf\_sub\_nr(i)) url2]);

consurf\_output(i)={cons};

else

consurf\_output(i)={'FAILED'};

end

end

consurf\_final=cell(286,1000);

progressbar('Final')

for i=1:n

progressbar(i/n)

sajz=size(seq{i});

gdzie2=consurf\_output{i};

fail=strcmp(gdzie2,'FAILED');

gdzie3=strfind(gdzie2,'\*Below');

if fail<1

gdzie4=gdzie2(1209:gdzie3(1)-5);

gdzie5=regexp(gdzie4,'\n','split');

for j=1:sajz(2)

gdzie6=gdzie5{j};

consurf\_final(i,j)={gdzie6(11:17)};

end

else

consurf\_final(i,1)={'FAILED'};

end

end

end

%PCA projection

S1=size(thaliana\_expression);

S2=size(PCA\_components);

AT\_dot\_prod=zeros(S(1),S2(2));

for i=1:S1(1)

for j=1:S2(2)

AT\_dot\_prod(i,j)=dot(AT\_expression(i,:),PCA\_components(:,j));

end

end

scatter(AT\_dot\_prod(:,1),AT\_dot\_prod(:,2),'b')

%ICA projection

S1=size(thaliana\_expression);

S2=size(ICA\_components);

AT\_dot\_prod=zeros(S(1),S2(2));

for i=1:S1(1)

for j=1:S2(2)

AT\_dot\_prod(i,j)=dot(AT\_expression(i,:),ICA\_components(:,j));

end

end

scatter(AT\_dot\_prod(:,1),AT\_dot\_prod(:,2),'b')

function MI=FastMI(data,s)

% data : the input data, rows correspond to genes columns correspond to arrays (samples)

% s : the std of the Gaussian kernel for density estimation

MI = zeros(size(data,1));

s\_square = s^2;

L = size(data,2);

for i=1:L

temp = data - repmat(data(:,i),1,L);

temp = exp(-(temp.^2)/(2\*s\_square));

temp1 = sum(temp,2);

temp2 = tmp\*temp';

for j=1:size(temp2,1)

temp2(j,:) = temp2(j,:)./temp1(j);

temp2(:,j) = temp2(:,j)./temp1(j);

end

MI = MI + log(temp2);

clear temp2

end

MI = MI/L + log(L);

%PCA on tree data objects, chapter 4, section 4.25.

model = classRF\_train(ELEMENT\_MATRIX4,ELEMENT\_MATRIX4\_CAT,1000);

model.ndbigtree = reshape(model.ndbigtree, size(model.ndbigtree,1) \* size(model.ndbigtree,2),1);

doPCA=zeros((449\*449-449)/2,1000);

for j=1:1000

j

tree\_num=j;

sztree = model.ndbigtree(1:model.ntree);

num\_nodes = sztree(tree\_num);

treemap = [model.treemap(:,tree\_num\*2-1); model.treemap(:,tree\_num\*2);];

lDau = treemap(1:2:end); lDau = lDau(1:num\_nodes);

rDau = treemap(2:2:end); rDau = rDau(1:num\_nodes);

nodestatus = model.nodestatus(1:num\_nodes,tree\_num);

nodeclass = model.nodeclass(1:num\_nodes,tree\_num);

bestvar = model.bestvar(1:num\_nodes,tree\_num);

xbestsplit = model.xbestsplit(1:num\_nodes,tree\_num);

DAR1=cell(length(nodestatus),1);

DAR2=zeros(length(nodestatus),1);

DAR3=zeros(length(nodestatus),1);

for i=1:length(nodestatus)

if nodestatus(i) ~= -1

DAR1(i)=MOTYWY\_ELEMENT(bestvar((i)));

DAR2(i)=i;

DAR3(i)=bestvar(i);

else

end

end

DRZEWO=zeros(449,449);

for i=1:length(nodestatus)

if nodestatus(i) ~= -1

komp1=strcmp(DAR1,char(MOTYWY\_ELEMENT(bestvar(i))));

komp2=strcmp(DAR1,char(MOTYWY\_ELEMENT(DAR3(DAR2==lDau(i)))));

komp3=strcmp(DAR1,char(MOTYWY\_ELEMENT(DAR3(DAR2==rDau(i)))));

DRZEWO(DAR3(komp1),DAR3(komp2))=1;

DRZEWO(DAR3(komp2),DAR3(komp1))=1;

DRZEWO(DAR3(komp1),DAR3(komp3))=1;

DRZEWO(DAR3(komp3),DAR3(komp1))=1;

end

end

c=1;

d=448;

e=1;

f=448;

WEKTOR=zeros((449\*449-449)/2,1);

for i=1:449

WEKTOR(e:d)=DRZEWO(c,c+1:end);

c=c+1;

e=e+f;

f=f-1;

d=d+f;

end

doPCA(:,j)=WEKTOR;

end

PAIR\_LABELS=cell((449\*449-449)/2,1);

c=1;

for i=1:449

for j=1:449

if j>i

PAIR\_LABELS(c,1)={[MOTYWY\_ELEMENT{j,1} '/' MOTYWY\_ELEMENT{i,1}]};

c=c+1;

end

end

end