######## INFORMATION ########

This software was proposed and supervised by Vladimir A. Kuznetsov. Initial codes were written by Efthimios Motakis.

Tang Zhiqun and Ow Ghim Siong made code's corrections and provided final versions of the codes and the examples.

the codes were improved after its testing by Tang Zhiqun, Ow Ghim Siong, Anna V. Ivshina and Vladimir A. Kuznetsov.

#Bioinformatics Institute, A-STAR;

#Date: 03. 07.2016

This folder contains functions and scripts, as well as example input files for

the following analyses:

1) one-dimensional data-driven grouping

2) two-dimensional data-driven grouping

3) voting grouping analysis

4) plotting of voting groups

For checking, the example output files are located in the folder /example_output.

The outputs for the individual script files are placed in their own folders.

######## REQUIRES ##########

R program + survival package

Python program + matplotlib + pandas + lifelines packages

######## CONTAINS ##########

This folder contains the following:

- Codes:
  - 1D.r (functions for 1D-DDg)
  - 2D.r (functions for 2D-DDg)
  - script_1ddg_and_2ddg.r (script to run 1D-DDg and 2D-DDg)
  - script_2D_voting.r (function and script to run voting analysis)
  - plot_voting_results.py (script to plot the voting survival curves)

- Example file:
  - exp.txt (expression data and clinical information of genes and samples.

The first columns contains the gene names and the header

contains the sampleids.

The second row contains the event clinical information.

The third row contains the time of event.

The gene intensity values begins from the fourth row.

- gene_pair.txt (contains the gene pairs information.

Each row contains the gene pairs)

- Example output file:

- 1D-DDg_results.txt and 1D-DDg_plots.pdf for 1D-DDg analysis.

- 2D-DDg_results.txt and 2D-DDg_plots.pdf for 2D-DDg analysis.

- voting_input.txt (created automatically by the script, for input to script_2D_voting.r)

- g2.txt, g3.txt and <___>.voting_plot.pdf for voting analysis (by script_2D_voting.r)

INSTRUCTIONS

############ In R programming environment ############

Set the working directory

> setwd(current_directory)

Run 1D-DDg and 2D-DDg analyses

Input - exp.txt

Output - 1D-DDg_results.txt,

- 1D-DDg_plots.pdf,

- 2D-DDg_results.txt,

- 2D-DDg_plots.pdf and

- voting_input.txt.

> source('script_1ddg_and_2ddg.r')

Run the voting analysis

Input - voting_input.txt (via the 'script_1ddg_and_2ddg.r')

Output - g2.txt (classification results for 2 groups)

- g3.txt (classification results for 3 groups)

- final.txt (do not use)

```
> source('script_2D_voting.r')


############  In Python programming environment ############

# Plot the voting classification

# Input  - g2.txt OR g3.txt

        - voting_input.txt

# Output - <g2.txt/g3.txt>.voting_plot.pdf

> python plot_voting_results.py


### END OF README  ###

-----------------------------------------
```

1D.R

```
---------------------------------------

library(survival)

#x[q,],t,e,20,title[q],doplot=TRUE

Data_driven_1DDD<-function(genedatum,t,e,patss,title,doplot){    ### ints is vector with patients

 ss<-quantile(genedatum,seq(0,1,0.05))

 range<-genedatum[genedatum>=ss[[2]]  & genedatum<=ss[[20]]]

 res.pv.alt<-rep(1,length(range))

 group<-matrix(1,length(range),length(t))

 for(j in 1:length(range)){

    for(k in 1:length(t)){

      if(genedatum[k]>range[j]){  group[j,k]<-2   }

    }

    group1<-group[j,]

    if(length(group1[group1==1])>patss & length(group1[group1==2])>patss){

#         res.pv.alt[j]<-signif(summary(coxph(Surv(t,e)~as.factor(group[j,])))[[6]][5],digits=500)


        cox =  coxph(Surv(t,e)~ as.factor(group1) );

        tem  = length( cox$coefficients [!is.na( cox$coefficients ) ])

        res.pv.alt[j]<-1-pchisq(cox$wald.test,df=tem) # wald.test P

        if ( res.pv.alt[j]==0)

         res.pv.alt[j] =  -expm1(pchisq(cox$wald.test,df=tem, log.p=T) )
```

```
        }
  }
  group2<-newgroups(t,group)
  design2<-rep(2,nrow(group2))
  for(i in 1:nrow(group2)){
     if(sum(group2[i,genedatum>range[i]])==length(group2[i,genedatum>range[i]])){
        design2[i]<-1
     }
  }

  opt.pval1<-min(res.pv.alt)[1]
  opt.group<-group[which(res.pv.alt==min(res.pv.alt))[1],]
  opt.group<-c(newgroups(t,matrix(opt.group,nrow=1)))
  opt.cutoff1<-range[which(res.pv.alt==min(res.pv.alt))[1]]
  res1<-c(opt.cutoff1,opt.pval1)
  resall<-matrix(c(range,res.pv.alt),ncol=2)

  sl<-sort.list(range)
  slrange<-range[sl]
  allp<-resall[sl,]
  wh<-which(slrange==res1[1])

  if(doplot==TRUE){
     par(mfrow=c(2,2))
     if(mean(t[genedatum<=res1[1]])>=mean(t[genedatum>=res1[1]])){

plot(slrange[1:wh],log(allp[1:wh,2]),type="l",xlim=c(min(slrange),max(slrange)),ylim=c(min(log(allp[,2])),max(log(allp[,2]))),main=paste(titl
e),xlab="cut-offs",ylab="log p-value")
        lines(slrange[(wh):length(slrange)],log(allp[(wh):length(slrange),2]),col=2)
     }
     if(mean(t[genedatum<=res1[1]])<=mean(t[genedatum>=res1[1]])){

plot(slrange[1:wh],log(allp[1:wh,2]),type="l",col=2,xlim=c(min(slrange),max(slrange)),ylim=c(min(log(allp[,2])),max(log(allp[,2]) )),main=pas
te(title),xlab="cut-offs",ylab="log p-value")
        lines(slrange[(wh):length(slrange)],log(allp[(wh):length(slrange),2]))
```

```
    }
    lines(-20:20,rep(log(res1[2]),41),lty=2)
    points(res1[1],log(res1[2]),pch=4,cex=1.5)
    plot(survfit(Surv(t, e)~ as.factor(opt.group)),col=c(1,2),main=paste(title),xlab="DFS(years)",ylab="Probability")
  }

  design<-2
  if(sum(opt.group[genedatum>opt.cutoff1])==length(opt.group[genedatum>opt.cutoff1])){
    design<-1
  }

  prop.test<-cox.zph(coxph(Surv(t,e) ~as.factor(opt.group)))[[1]][3]
  npg<-c(length(opt.group[opt.group==1]),length(opt.group[opt.group==2]))

  time1 = mean( t[opt.group==1 & e ==1] );
  time2 = mean( t[opt.group==2 & e ==1] );
  time = c(time1, time2);
  return(list(res1,range,opt.group,resall,design,design2,prop.test,npg, time))
}


newgroups<-function(t,g){ ###1=LOW-RISK, 2=HIGH-RISK, 3=MEDIUM-RISK1 4=MEDIUM-RISK2 ###
  groups.new<-matrix(0,nrow(g),ncol(g))
  for(i in 1:nrow(groups.new)){
    mm<-rep(-999,max(g[i,]))
    for(j in 1:max(g[i,])){
      if(length(g[i,][g[i,]==j])>0){  mm[j]<-mean(t[g[i,]==j])  }
    }
    gg<-rep(0,length(t))
    sl<-sort.list(mm,decreasing=TRUE)
    for(j in 1:length(sl)){
      gg[g[i,]==sl[j]]<-j
    }
```

```
    groups.new[i,]<-gg
  }
  return(groups.new)
}
```

2D.R

```
Data_driven_2DDD<-function(a1,a2,cut1,cut2,t,e,patss, title){

  pats  = length(t)
  stats = rep(0,6)

  group = rep(1,pats)
  for(i in 1:pats){
    if(a1[i] >cut1){   group[i] = 2    }
  }

  ss1 =  1-pchisq(coxph(Surv(t, e)~ as.factor(group))$wald.test,1)
  stats[1] = ss1

  group = rep(1,pats)
  for(i in 1:pats){
    if(a2[i] >cut2){ group[i] = 2   }
  }
  ss2 = 1-pchisq(coxph(Surv(t, e)~ as.factor(group))$wald.test,1)
  stats[2] = ss2
  stats[3:4] = c(cut1,cut2)

  data<-matrix(cbind(matrix(c(a1,a2),length(a1),2),rep(0,length(a1))),ncol=3)
  data[ data[,1] <  cut1 & data[,2] <  cut2,3 ] = 1
  data[ data[,1] >= cut1 & data[,2] <  cut2,3 ] = 2
  data[ data[,1] <  cut1 & data[,2] >= cut2,3 ] = 3
  data[ data[,1] >= cut1 & data[,2] >= cut2,3 ] = 4

  group1<-c(data[,3])
```

```
g1t<-t[group1==1]

g1e<-e[group1==1]

g1g<-group1[group1==1]


g2t<-t[group1==2]

g2e<-e[group1==2]

g2g<-group1[group1==2]


g3t<-t[group1==3]

g3e<-e[group1==3]

g3g<-group1[group1==3]


g4t<-t[group1==4]

g4e<-e[group1==4]

g4g<-group1[group1==4]


tt12<-c(g1t,g2t)

ee12<-c(g1e,g2e)

gg12<-c(g1g,g2g)

tt13<-c(g1t,g3t)

ee13<-c(g1e,g3e)

gg13<-c(g1g,g3g)

tt14<-c(g1t,g4t)

ee14<-c(g1e,g4e)

gg14<-c(g1g,g4g)

tt23<-c(g2t,g3t)

ee23<-c(g2e,g3e)

gg23<-c(g2g,g3g)

tt24<-c(g2t,g4t)

ee24<-c(g2e,g4e)

gg24<-c(g2g,g4g)

tt34<-c(g3t,g4t)

ee34<-c(g3e,g4e)
```

```
gg34<-c(g3g,g4g)


res2g<-rep(0,5)
gr2<-matrix(0,5,2)


gr1.1<-rep(1,length(tt14))
gr2.1<-rep(2,length(tt23))
if(length(gr1.1)>patss & length(gr2.1)>patss){
   res2g[1]<-1-pchisq(coxph(Surv(c(tt14,tt23), c(ee14,ee23))~ as.factor(c(gr1.1,gr2.1)))$wald.test,1)
}
if(length(gr1.1)<(patss+1) | length(gr2.1)<(patss+1)){  res2g[1]<-1   }


gr1.2<-rep(1,(length(tt12)+length(g3t)))
gr2.2<-rep(2,length(g4t))
if(length(gr1.2)>patss & length(gr2.2)>patss){
  res2g[2]<-1-pchisq(coxph(Surv(c(tt12,g3t,g4t), c(ee12,g3e,g4e))~ as.factor(c(gr1.2,gr2.2)))$wald.test,1)
}
if(length(gr1.2)<(patss+1) | length(gr2.2)<(patss+1)){ res2g[2]<-1 }


gr1.3<-rep(1,(length(tt13)+length(g4t)))
gr2.3<-rep(2,length(g2t))
if(length(gr1.3)>patss & length(gr2.3)>patss){
  res2g[3]<-1-pchisq(coxph(Surv(c(tt13,g4t,g2t), c(ee13,g4e,g2e))~ as.factor(c(gr1.3,gr2.3)))$wald.test,1)
}
if(length(gr1.3)<(patss+1) | length(gr2.3)<(patss+1)){ res2g[3]<-1   }


gr1.4<-rep(1,(length(tt23)+length(g4t)))
gr2.4<-rep(2,length(g1t))
if(length(gr1.4)>patss & length(gr2.4)>patss){
   res2g[4]<-1-pchisq(coxph(Surv(c(tt23,g4t,g1t), c(ee23,g4e,g1e))~ as.factor(c(gr1.4,gr2.4)))$wald.test,1)
}
if(length(gr1.4)<(patss+1) | length(gr2.4)<(patss+1)){ res2g[4]<-1   }
```

```
gr1.5<-rep(1,(length(tt12)+length(g4t)))

gr2.5<-rep(2,length(g3t))

if(length(gr1.5)>patss & length(gr2.5)>patss){

   res2g[5]<-1-pchisq(coxph(Surv(c(tt12,g4t,g3t), c(ee12,g4e,g3e))~  as.factor(c(gr1.5,gr2.5)))$wald.test,1)

}

if(length(gr1.5)<(patss+1) | length(gr2.5)<(patss+1)){  res2g[5]<-1  }


gr2[1,]<-c(length(gr1.1),length(gr2.1))

gr2[2,]<-c(length(gr1.2),length(gr2.2))

gr2[3,]<-c(length(gr1.3),length(gr2.3))

gr2[4,]<-c(length(gr1.4),length(gr2.4))

gr2[5,]<-c(length(gr1.5),length(gr2.5))




which.l<-which(res2g==min(res2g))

gresults<-c(1,2,which.l[1])

presults<-res2g


res.fin1<-c(1,2,which.l[1],presults[gresults[3]])


groups<-rep(1,length(t))

if(res.fin1[3]==1){

   groups[a1<=cut1 & a2<=cut2]<-2

   groups[a1>cut1 & a2>cut2]<-2

}

if(res.fin1[3]==2){

   groups[a1>cut1 & a2>cut2]<-2

}

if(res.fin1[3]==3){

   groups[a1>cut1 & a2<=cut2]<-2

}

if(res.fin1[3]==4){

   groups[a1<=cut1 & a2<=cut2]<-2
```

```
}
if(res.fin1[3]==5){
   groups[a1<=cut1 & a2>cut2]<-2
}
groups<-c(newgroups(t,matrix(groups,nrow=1)))
stats[5:6]<-res.fin1[3:4]

if(stats[5]==1){
   if(sum(groups[a1>cut1 & a2>cut2])==length(groups[a1>cut1 & a2>cut2])){
      stats[5]<-1.1
   }
   if(sum(groups[a1>cut1 & a2>cut2])!=length(groups[a1>cut1 & a2>cut2])){
      stats[5]<-1.2
   }
}
if(stats[5]==2){
   if(sum(groups[a1>cut1 & a2>cut2])==length(groups[a1>cut1 & a2>cut2])){
     stats[5]<-2.1
   }
   if(sum(groups[a1>cut1 & a2>cut2])!=length(groups[a1>cut1 & a2>cut2])){
      stats[5]<-2.2
   }
}
if(stats[5]==3){
   if(sum(groups[a1>cut1 & a2>cut2])==length(groups[a1>cut1 & a2>cut2])){
     stats[5]<-3.1
   }
   if(sum(groups[a1>cut1 & a2>cut2])!=length(groups[a1>cut1 & a2>cut2])){
      stats[5]<-3.2
   }
}
if(stats[5]==4){
   if(sum(groups[a1>cut1 & a2>cut2])==length(groups[a1>cut1 & a2>cut2])){
```

```
      stats[5]<-4.1
    }
    if(sum(groups[a1>cut1 & a2>cut2])!=length(groups[a1>cut1 & a2>cut2])){
      stats[5]<-4.2
    }
  }
  if(stats[5]==5){
    if(sum(groups[a1>cut1 & a2>cut2])==length(groups[a1>cut1 & a2>cut2])){
      stats[5]<-5.1
    }
    if(sum(groups[a1>cut1 & a2>cut2])!=length(groups[a1>cut1 & a2>cut2])){
      stats[5]<-5.2
    }
  }

   plot(survfit(Surv(t, e)~ as.factor(groups)),col=c(1,2),main=paste(title),xlab="Time(years)",ylab="Survival
Probability", cex.axis=1.5, cex.lab=1.5, cex.main=2)
   title2 = strsplit(title, ":")[[1]];
   plot(a1,a2, col=groups,xlab=title2[1],ylab=title2[2], main="scatterplot", cex.axis=1.5, cex.lab=1.5, cex=0.8,
cex.main=2)
   abline(v=cut1, lty=2, col="black",lwd=3);
   abline(h=cut2, lty=2, col="black",lwd=3);
   legend("topleft",c("goog-prognosis", "poor-prognosis"), pch=21, col=c(1,2), cex=0.8)

   prop.test<-cox.zph(coxph(Surv(t,e) ~as.factor(groups)))[[1]][3]
   npg<-c(length(groups[groups==1]),length(groups[groups==2]))
  #  sink("tmp");
  #  print(survfit(Surv(t, e)~ as.factor(groups)), rmean="individual")
  #  sink();
  #  b =scan("tmp", what="character")

    time1 = mean( t[groups==1 & e ==1] );
    time2 = mean( t[groups==2 & e ==1] );
```

```
    time = c(time1, time2);


 #  area = c(as.numeric(b[20]), as.numeric(b[28]) )
   return(list(stats,groups,prop.test,npg, time))
}




newgroups<-function(t,g){ ###1=LOW-RISK, 2=HIGH-RISK, 3=MEDIUM-RISK1 4=MEDIUM-RISK2 ###


 groups.new<-matrix(0,nrow(g),ncol(g))
 for(i in 1:nrow(groups.new)){
    mm<-rep(-999,max(g[i,]))
    for(j in 1:max(g[i,])){
       if(length(g[i,][g[i,]==j])>0){
       mm[j]<-mean(t[g[i,]==j])
       }
    }
    gg<-rep(0,length(t))
    sl<-sort.list(mm,decreasing=TRUE)
    for(j in 1:length(sl)){   gg[g[i,]==sl[j]]<-j    }
    groups.new[i,]<-gg
 }
 return(groups.new)
}
```

Script_1ddg_and _2ddg.R

```
#========  main function
source("1D.r")
source("2D.r")


data = read.table("exp.txt",header =T, row.names=1);
```

```
pairs =as.matrix(read.table("gene_pair.txt", header=T) );

#################### ONE-DIMENSIONAL DATA-DRIVEN GROUPING ####################

id = colnames(data);# patient sample ID
genes =rownames(data) # gene names
genes =genes[-(1:2)]

e= data[1,]; # event ( 0=alive, 1=dead)
e=as.numeric(e)
names(e)=id;

t= data[2,]; #time
t= as.numeric(as.character(t) )
names(t)=id;

data = data[-c(1:2),] # expression data
colnames(data)=id;
rownames(data)=genes;

title = unique(c(pairs[,1], pairs[,2]) );
list1<-title
mm1<-match(list1,genes)

x<-as.matrix(data[mm1,]);

resDDp<-matrix(0,length(mm1),2)
design<-rep(0,length(mm1))
resDDm<-matrix(0,length(mm1),2)
resDDw<-rep(0,length(mm1))
resDDpat<-matrix(0,length(mm1),length(t))
ptest<-rep(0,length(mm1))
npg<-matrix(0,length(mm1),2)
```

```
pdf("1D-DDg_plots.pdf")

par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))

for(q in 1:length(mm1)){

 print(paste('1DDg for gene', q))

 res<-Data_driven_1DDD(x[q,],t,e,35,title[q],doplot=TRUE)

 resDDp[q,]<-res[[1]]

 resDDpat[q,]<-res[[3]]

 design[q]<-res[[5]]

 resDDm[q,]<-c(mean(x[q,resDDpat[q,]==1]),mean(x[q,resDDpat[q,]==2]))

 resDDw[q]<-wilcox.test(x[q,resDDpat[q,]==1],x[q,resDDpat[q,]==2])$p.value

 ptest[q]<-res[[7]]

 npg[q,]<-res[[8]]

}

dev.off()


# set colnames

colnames(resDDp) <- c('Pvalue', 'cutoff')

colnames(resDDm) <- c("mean1", "mean2")

colnames(npg) <- c('N1', 'N2')

colnames(resDDpat) <- id


# FORMAT 1DDG Output

DD_1D = cbind(genes = list1, resDDp, design, resDDm, wilcox=resDDw, proptest=ptest, npg, resDDpat)


# SAVE TO FILE

write.table(DD_1D, file="1D-DDg_results.txt", sep="\t",quote=F, row.names=F);



#################### TWO-DIMENSIONAL DATA-DRIVEN GROUPING ######################

c = resDDp[,1];

data=x;

genes = rownames(data);
```

```
cut =c


l1<-as.character(pairs[,1])

l2<-as.character(pairs[,2])

mm1<-match(l1,genes)

mm2<-match(l2,genes)

title=paste(l1,l2, sep=":")


res.p<-matrix(0,length(l1),6)

res.pat<-matrix(0,length(l1),length(t))

resDDm<-matrix(0,length(mm1),6)

resDDw<-rep(0,length(mm1))

ptest<-rep(0,length(mm1))

npg<-matrix(0,length(mm1),2)

ar<-matrix(0,length(mm1),2)

ntime <-matrix(0,length(mm1),2)


pdf("2D-DDg_plots.pdf")

par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))

for(k in 1:length(l1)){

 print(paste('2DDg for gene pair', k))

  ints1<-as.matrix(data[mm1[k],])

  ints2<-as.matrix(data[mm2[k],])

  cuts1<-cut[mm1[k]]

  cuts2<-cut[mm2[k]]

  res<-Data_driven_2DDD(ints1,ints2,cuts1,cuts2,t,e,20, title[k])

  res.p[k,]<-res[[1]]

  res.pat[k,]<-res[[2]]

  ptest[k]<-res[[3]]

  ints<-matrix(c(ints1,ints2),ncol=2)

  a1 =  mean(ints[res.pat[k,]==1,1])

  a2 =  mean(ints[res.pat[k,]==2,1])

  b1 =  mean(ints[res.pat[k,]==1,2])
```

```
   b2 =   mean(ints[res.pat[k,]==2,2])
   resDDm[k,]<-c(a1,a2,a1/a2,b1,b2,b1/b2)
   resDDw[k]<-wilcox.test(ints[res.pat[k,]==1,],ints[res.pat[k,]==2,])$p.value
   npg[k,]<-res[[4]]
  # ar[k,] <-res[[5]];
   ntime[k,] <- res[[5]];
}
dev.off();

# Set column names
colnames(res.p) <- c('pv1', 'pv2', 'cutoff1', 'cutoff2', 'design', '2Dpvalue')
colnames(resDDm) <- c('mean_of_gene1_in_cluster1',
                                                    'mean_of_gene1_in_cluster2',
                                                    'fold_change_of_gene1',
                                                    'mean_of_gene2_in_cluster1',
                                                    'mean_of_gene2_in_cluster2',
                                                    'fold_change_of_gene2')
colnames(ntime) <- c('mean_survival_time_of_cluster1','mean_survival_time_of_cluster2')
colnames(npg) <- c('number_of_lo', 'number_of_hi')
colnames(res.pat) <- id

# FORMAT 2DDG Output
DD_2D = cbind(genes = title, res.p, fdr=p.adjust(res.p[,6],"BY"),resDDm, ntime, resDDw, ptest,npg, res.pat)

 # SAVE TO FILE
write.table(DD_2D, file="2D-DDg_results.txt", sep="\t",quote=F,  row.names=F);



#####################  PREPARATION OF FILE FOR VOTING ######################

# SORT by pvalue in ascending value
ordered_DD_2D = DD_2D[sort.list(DD_2D[,'2Dpvalue']),]
#write.table(ordered_DD_2D, file="ordered_2D-DDg_results.txt", sep="\t",quote=F,  row.names=F);
```

```
# format to the required input to 2D_voting.R

voting_input <- ordered_DD_2D

tmp <- strsplit(voting_input[,'genes'], ":", fixed = TRUE)

voting_input <- cbind(gene1=sapply(tmp, "[[", 1),
                                                    gene2=sapply(tmp, "[[", 2),
                                                    voting_input)


# Drop columns that are not needed.

drop_cols = c('genes', 'pv1', 'pv2', 'cutoff1', 'cutoff2', 'design', 'fdr',
                                        'mean_of_gene1_in_cluster1',
                                        'mean_of_gene1_in_cluster2',
                                        'fold_change_of_gene1',
                                        'mean_of_gene2_in_cluster1',
                                        'mean_of_gene2_in_cluster2',
                                        'fold_change_of_gene2',
                                        'mean_survival_time_of_cluster1',
                                        'mean_survival_time_of_cluster2',
                                        'resDDw', 'ptest',
                                        'number_of_lo', 'number_of_hi')

voting_input <- voting_input[ , !(colnames(voting_input) %in% drop_cols)]


# add in the time and event loaded at the begninning.

voting_input <- rbind(c('na','na','na',e),
                                                    c('na','na', 'na', t),
                                                    voting_input)


# write to file, to be used by 2D_voting.R.

write.table(voting_input, file="voting_input.txt", sep="\t",quote=F,  row.names=F, col.names=F);
```

---

script_2D_voting.R

```
---------------------------------


library(survival)


voting <-function(t,e,datatop,data,genes,cut,id,groups,groups_def,patss){


 list1<-datatop[,1]
 list2<-datatop[,2]
 mm1<-match(list1,genes)
 mm2<-match(list2,genes)
 resp2<-matrix(0,nrow(datatop),3)
 resp3<-matrix(0,nrow(datatop),5)
 respat<-matrix(0,nrow(datatop),length(t))
 fgroups2<-matrix(0,nrow(datatop),length(t))
 fgroups3<-matrix(0,nrow(datatop),length(t))
 npats2<-matrix(0,nrow(datatop),2)
 npats3<-matrix(0,nrow(datatop),3)
 for(q in 1:nrow(datatop)){
  print(q);
   if(q==1){
      if(groups_def=="F"){
         if(id[q]=="2D"){
            ints1<-data[mm1[q],]
            ints2<-data[mm2[q],]
            cuts1<-cut[mm1[q]]
            cuts2<-cut[mm2[q]]
            dd<-DDalg2D(ints1,ints2,cuts1,cuts2,t,e,patss)
            resp2[q,]<-c(q,dd[[1]][6],dd[[1]][6])
            resp3[q,]<-c(q,dd[[1]][6],rep(1,3))
            respat[q,]<-dd[[2]]
            fgroups2[q,]<-respat[q,]
            fgroups3[q,]<-respat[q,]
            npats2[q,]<-c(length(fgroups2[q,][fgroups2[q,]==1]),length(fgroups2[q,][fgroups2[q,]==2]))
```

```
      npats3[q,]<-c(length(fgroups3[q,][fgroups3[q,]==1]),length(fgroups3[q,][fgroups3[q,]==2]),0)
    }
    if(id[q]=="1D"){
      ints1<-data[mm1[q],]
      dd<-DDalg1D(ints1,t,e,patss,"",doplot=FALSE)
      resp2[q,]<-c(q,dd[[1]][2],dd[[1]][2])
      resp3[q,]<-c(q,dd[[1]][2],rep(1,3))
      respat[q,]<-dd[[3]]
      fgroups2[q,]<-respat[q,]
      fgroups3[q,]<-respat[q,]
      npats2[q,]<-c(length(fgroups2[q,][fgroups2[q,]==1]),length(fgroups2[q,][fgroups2[q,]==2]))
      npats3[q,]<-c(length(fgroups3[q,][fgroups3[q,]==1]),length(fgroups3[q,][fgroups3[q,]==2]),0)
    }
    if(id[q]=="Ratio"){
      ints1<-data[mm1[q],]
      ints2<-data[mm2[q],]
      dd<-DDalg1D((ints1-ints2),t,e,patss,"",doplot=FALSE)
      resp2[q,]<-c(q,dd[[1]][2],dd[[1]][2])
      resp3[q,]<-c(q,dd[[1]][2],rep(1,3))
      respat[q,]<-dd[[3]]
      fgroups2[q,]<-respat[q,]
      fgroups3[q,]<-respat[q,]
      npats2[q,]<-c(length(fgroups2[q,][fgroups2[q,]==1]),length(fgroups2[q,][fgroups2[q,]==2]))
      npats3[q,]<-c(length(fgroups3[q,][fgroups3[q,]==1]),length(fgroups3[q,][fgroups3[q,]==2]),0)
    }
  }
  if(groups_def=="T"){
    if(id[q]=="2D"){
      resp2[q,]<-c(q,groups[q,1],groups[q,1])
      resp3[q,]<-c(q,groups[q,1],rep(1,3))
      respat[q,]<-groups[q,2:ncol(groups)]
      fgroups2[q,]<-respat[q,]
      fgroups3[q,]<-respat[q,]
```

```
      npats2[q,]<-c(length(fgroups2[q,][fgroups2[q,]==1]),length(fgroups2[q,][fgroups2[q,]==2]))
      npats3[q,]<-c(length(fgroups3[q,][fgroups3[q,]==1]),length(fgroups3[q,][fgroups3[q,]==2]),0)
    }
    if(id[q]=="1D"){
      resp2[q,]<-c(q,groups[q,1],groups[q,1])
      resp3[q,]<-c(q,groups[q,1],rep(1,3))
      respat[q,]<-groups[q,2:ncol(groups)]
      fgroups2[q,]<-respat[q,]
      fgroups3[q,]<-respat[q,]
      npats2[q,]<-c(length(fgroups2[q,][fgroups2[q,]==1]),length(fgroups2[q,][fgroups2[q,]==2]))
      npats3[q,]<-c(length(fgroups3[q,][fgroups3[q,]==1]),length(fgroups3[q,][fgroups3[q,]==2]),0)
    }
    if(id[q]=="Ratio"){
      resp2[q,]<-c(q,groups[q,1],groups[q,1])
      resp3[q,]<-c(q,groups[q,1],rep(1,3))
      respat[q,]<-groups[q,2:ncol(groups)]
      fgroups2[q,]<-respat[q,]
      fgroups3[q,]<-respat[q,]
      npats2[q,]<-c(length(fgroups2[q,][fgroups2[q,]==1]),length(fgroups2[q,][fgroups2[q,]==2]))
      npats3[q,]<-c(length(fgroups3[q,][fgroups3[q,]==1]),length(fgroups3[q,][fgroups3[q,]==2]),0)
    }
  }
}
if(q>1){
  if(groups_def=="F"){
    if(id[q]=="2D"){
      ints1<-data[mm1[q],]
      ints2<-data[mm2[q],]
      cuts1<-cut[mm1[q]]
      cuts2<-cut[mm2[q]]
      dd<-DDalg2D(ints1,ints2,cuts1,cuts2,t,e,patss)
      resp2[q,1:2]<-c(q,dd[[1]][6])
      resp3[q,1:2]<-c(q,dd[[1]][6])
```

```
    respat[q,]<-dd[[2]]
  }
  if(id[q]=="1D"){
    ints1<-data[mm1[q],]
    dd<-DDalg1D(ints1,t,e,patss,"",doplot=FALSE)
    resp2[q,]<-c(q,dd[[1]][2])
    resp3[q,]<-c(q,dd[[1]][2])
    respat[q,]<-dd[[3]]
  }
  if(id[q]=="Ratio"){
    ints1<-data[mm1[q],]
    ints2<-data[mm1[q],]
    dd<-DDalg1D((ints1-ints2),t,e,patss,"",doplot=FALSE)
    resp2[q,]<-c(q,dd[[1]][2])
    resp3[q,]<-c(q,dd[[1]][2])
    respat[q,]<-dd[[3]]
  }
  }
if(groups_def=="T"){
  if(id[q]=="2D"){
    resp2[q,1:2]<-c(q,groups[q,1])
    resp3[q,1:2]<-c(q,groups[q,1])
    respat[q,]<-groups[q,2:ncol(groups)]
  }
  if(id[q]=="1D"){
    resp2[q,1:2]<-c(q,groups[q,1])
    resp3[q,1:2]<-c(q,groups[q,1])
    respat[q,]<-groups[q,2:ncol(groups)]
  }
  if(id[q]=="Ratio"){
    resp2[q,1:2]<-c(q,groups[q,1])
    resp3[q,1:2]<-c(q,groups[q,1])
    respat[q,]<-groups[q,2:ncol(groups)]
```

```
            }
        }
        weights<-(-log(resp2[1:q,2]))/sum(-log(resp2[1:q,2]))
        groups2<-respat[1:q,]
        groups2[groups2==2]<-0
        res<-weighted_groups2_alg(t,e,groups2,weights,0.2,0.8,0.01,patss)
        sl<-sort.list(res[,2])
        resp2[q,3]<-res[sl[1],2]
        fgroups2[q,]<-res[sl[1],5:ncol(res)]
        npats2[q,]<-res[sl[1],3:4]
        res<-weighted_groups3_alg(t,e,groups2,weights,0.44,0.2,0.01,patss)
        sl<-matrix(0,nrow(res),4)
        sl[,1]<-seq(1,nrow(res),1)
        sl1<-sort.list(res[,2])
        sl2<-sort.list(res[,3])
        sl3<-sort.list(res[,4])
        sl[,2]<-res[order(sl1),1]
        sl[,3]<-res[order(sl2),1]
        sl[,4]<-res[order(sl3),1]
        finsl<-apply(sl[,2:4],1,sum)
        best<-sl[which(finsl==min(finsl))[1],1]
        resp3[q,3:5]<-res[best,2:4]
        fgroups3[q,]<-res[best,8:ncol(res)]
        npats3[q,]<-res[best,5:7]
      }
    }
    groups2.stats<-matrix(cbind(resp2,npats2,fgroups2),ncol=(5+length(t)))
    groups3.stats<-matrix(cbind(resp3,npats3,fgroups3),ncol=(8+length(t)))
    write(t(groups2.stats),"g2.txt",ncolumns=ncol(groups2.stats),sep="\t")
    write(t(groups3.stats),"g3.txt",ncolumns=ncol(groups3.stats),sep="\t")
}
```

```
weighted_groups2_alg<-function(t,e,groups2,weights,startpoint,endpoint,cuts,patss){
 ss<-seq(startpoint,endpoint,cuts)
 groups<-matrix(1,length(ss),length(t))
 groupslength<-matrix(0,length(ss),2)
 result<-rep(1,length(ss))
 res<-matrix(0,nrow(groups2),ncol(groups2))
 for(i in 1:nrow(groups2)){
    for(j in 1:ncol(groups2)){
      res[i,j]<-weights[i]*groups2[i,j]
    }
 }
 res1<-apply(res,2,sum)

 for(i in 1:length(ss)){
    groups[i,res1<ss[i]]<-2
    groupslength[i,]<-c(length(groups[i,][groups[i,]==1]),length(groups[i,][groups[i,]==2]))
    if(groupslength[i,1]>=patss & groupslength[i,2]>=patss){
       cox =  coxph(Surv(t,e)~ as.factor(groups[i,]) );
       tem  = length( cox$coefficients [!is.na( cox$coefficients ) ])
       result[i]<-1-pchisq(cox$wald.test,df=tem) # wald.test P
       if (result[i]==0)
       result[i] =  -expm1(pchisq(cox$wald.test,df=tem, log.p=T) )
    }
    if(groupslength[i,1]<patss | groupslength[i,2]<patss){
       result[i]<-1
    }
 }
 final<-matrix(cbind(seq(1,length(result),1),result,groupslength,groups),ncol=(4+length(t)))
 write(t(final),"final.txt",ncolumns=ncol(final),sep="\t")
 return(final)
}

weighted_groups3_alg<-function(t,e,groups2,weights,startpoint,endpoint,cuts,patss){
```

```
 ss.down<-seq(startpoint,endpoint,-cuts)
 ss.up<-seq((1-startpoint),(1-endpoint),cuts)
 groups3<-matrix(2,(length(ss.up)*length(ss.down)),length(t))
 groupslength<-matrix(0,(length(ss.up)*length(ss.down)),3)
 result<-matrix(1,(length(ss.up)*length(ss.down)),3)
 res<-matrix(0,nrow(groups2),ncol(groups2))
 for(i in 1:nrow(groups2)){
    for(j in 1:ncol(groups2)){
      res[i,j]<-weights[i]*groups2[i,j]
    }
 }
 res1<-apply(res,2,sum)

 count<-0
 for(i in 1:length(ss.down)){
    for(j in 1:length(ss.up)){
       count<-count+1
       groups3[count,res1<ss.down[i]]<-3
       groups3[count,res1>ss.up[j]]<-1
       groupslength[count,]<-c(length(groups3[count,][groups3[count,]==1]),length(groups3[count,][groups3[count,]
==2]),length(groups3[count,][groups3[count,]==3]))
       if(groupslength[count,1]>=patss & groupslength[count,2]>=patss & groupslength[count,3]>=patss){
       #    result[count,]<-c(summary(coxph(Surv(t[groups3[count,]!=3], e[groups3[count,]!=3])~ as.factor(groups3
[count,][groups3[count,]!=3])))[[6]][5],summary(coxph(Surv(t[groups3[count,]!=2], e[groups3[count,]!=2])~ as.factor
(groups3[count,][groups3[count,]!=2])))[[6]][5],summary(coxph(Surv(t[groups3[count,]!=1], e[groups3[count,]!=1])~
as.factor(groups3[count,][groups3[count,]!=1])))[[6]][5])

        cox1 =  coxph(Surv(t[groups3[count,]!=3], e[groups3[count,]!=3])~ as.factor(groups3[count,][groups3
[count,]!=3]))
        tem1  = length( cox1$coefficients [!is.na( cox1$coefficients ) ])
        a <-1-pchisq(cox1$wald.test,df=tem1) # wald.test P
        if (a==0){ a =  -expm1(pchisq(cox1$wald.test,df=tem1, log.p=T) )   }
```

```
        cox2 =  coxph(Surv(t[groups3[count,]!=2], e[groups3[count,]!=2])~ as.factor(groups3[count,][groups3
[count,]!=2]))
        tem2  = length( cox2$coefficients [!is.na( cox2$coefficients ) ])
        b <-1-pchisq(cox2$wald.test,df=tem2) # wald.test P
        if (b==0)
        b =  -expm1(pchisq(cox2$wald.test,df=tem2, log.p=T) )

        cox3 =  coxph(Surv(t[groups3[count,]!=1], e[groups3[count,]!=1])~ as.factor(groups3[count,][groups3
[count,]!=1]))
        tem3  = length( cox3$coefficients [!is.na( cox3$coefficients ) ])
        c <-1-pchisq(cox3$wald.test,df=tem3) # wald.test P
        if (c==0)
        c =  -expm1(pchisq(cox3$wald.test,df=tem3, log.p=T) )

         result[count,]<-c(a,b,c);



       }
          if(groupslength[count,1]<patss | groupslength[count,2]<patss | groupslength[count,3]<patss){
             result[count,]<-c(1,1,1)
          }
    }
 }
 final<-matrix(cbind(seq(1,nrow(result),1),result,groupslength,groups3),ncol=(7+length(t)))
 write(t(final),"final.txt",ncolumns=ncol(final),sep="\t")
 return(final)
}



D=read.table("voting_input.txt")
datatop = D[,1:2]
datatop=datatop[-c(1:2),];
cut =NA
```

```
data=NA;

genes=NA;

id=rep("2D", length = dim(D)[1])

groups = as.matrix(D[,-(1:2)])

e=groups[1,]

e=as.numeric(e[-1]);

t=groups[2,]

t=as.numeric(t[-1]);

groups=groups[-c(1:2),]

groups = matrix(as.numeric(groups), ncol=dim(groups)[2], nrow=dim(groups)[1])


voting(t,e,datatop,data,genes,cut,id,groups,"T",20)
```

---

plot_voting_results.py

```
'''
This script plots the Kaplan-Meier classification after voting.


The input to this script as as follows:
   - voting_input.txt (for the time and event)
   - g2.txt (for the classification of 2 groups)
     or
     g3.txt (for the classification of 3 groups)



Requires the following library:
   - pandas
   - lifelines
   - matplotlib


Output file of this script:
   - voting_plot.pdf
'''
```

```
# These packages are required
import pandas as pd
import lifelines
import lifelines.statistics as stats
import matplotlib.pyplot as plt


if __name__ == "__main__":

    # PLEASE SET THESE.
    voting_input_fp = r"voting_input.txt"
    voting_output_fp = r"g3.txt"

    # Load
    voting_input_df = pd.read_csv(voting_input_fp, sep='\t',
                                  index_col=None, header=None)
    voting_output_df = pd.read_csv(voting_output_fp, sep='\t',
                                   index_col=0, header=None)

    # get clinical data from the first two rows, and third columns onwards
    clinical_df = voting_input_df.iloc[:2, 3:].T
    clinical_df.columns = ["Event", "Time"]
    clinical_df = clinical_df[["Time", "Event"]]

    # format the voting output
    # this is dependent on the input file, whether it is
    # g2.txt for 2 group classification
    # or g3.txt for 3 group classification
    if voting_output_df.loc[:, [3,4]].sum(axis=1).unique().shape[0] == 1:
        # columns 3 and 4 refers to the number of group 1 and group 2
        # for g2.txt.
        # if all the numbers across the rows are the same, this means
        # that this is g2.txt
```

```
    num_samples = voting_output_df.shape[1] - 4
    sampleids = ['sample%i' % (i+1) for i in xrange(num_samples)]
    columns = ['2Dpvalue', 'votingp', 'num1', 'num2'] + sampleids
    voting_output_df.columns = columns
    voting_output_df.index.name = "NumGenePairs"

elif voting_output_df.loc[:, [5,6,7]].sum(axis=1).unique().shape[0] == 1:
    # columns 5, 6, and 7 refers to the number of group 1, group 2 and group 3
    # for g3.txt.
    # if all the numbers across the rows are the same, this means
    # that this is g3.txt
    num_samples = voting_output_df.shape[1] - 7
    sampleids = ['sample%i' % (i+1) for i in xrange(num_samples)]
    columns = ['2Dpvalue', 'votingp1', 'votingp2', 'votingp3',
               'num1', 'num2', 'num3'] + sampleids
    voting_output_df.columns = columns
    voting_output_df.index.name = "NumGenePairs"

else:
    error = ("Invalid voting_output_fp. Set to g2.txt or g3.txt "
             "from 2D_voting.R")
    raise Exception, error



# format the classification data for plotting
max_num_genepair = voting_output_df.index.max()
groupname = 'num_features: %s' % max_num_genepair
clinical_df[groupname] = \
    voting_output_df.loc[max_num_genepair, sampleids].values
clinical_df.index = sampleids

# pvalue
multivariate_p = stats.multivariate_logrank_test(
```

```
    clinical_df['Time'], clinical_df[groupname],
    event_observed=clinical_df['Event']).p_value

# initialize figure axis
fig = plt.figure(1)
fig.clear()
fig.set_size_inches(5,4)
ax = fig.add_subplot(111)

# plot for each risk subgroup
subgroups = clinical_df[groupname].unique()
for subgrp in subgroups:
    subdf = clinical_df[clinical_df[groupname]==subgrp]
    kmf = lifelines.KaplanMeierFitter()
    kmf.fit(subdf['Time'], event_observed=subdf['Event'],
          label="riskgroup:%s" % str(int(subgrp)))
    kmf.plot(ax=ax, ci_show=False, flat=False, show_censors=True)

# format the figure
title = 'Classification via %s gene-pairs\nmultivariate logrank p:%.4g' % \
    (max_num_genepair, multivariate_p)
ax.set_title(title)
ax.set_ylabel("Proportion")
ax.set_xlabel("Time")
fig.tight_layout()

# save to file
fig.savefig(voting_output_fp+".voting_plot.pdf")
```