Exercise 5

Group C

26/1/2022

Transcriptional Regulations

For the first part we focus on trascriptional data. The idea is to try to find relations between genes regarding their transcription. We will implement some of the method we've seen in class.

Data preparation

```
library(infotheo)
library(ppcor)
set.seed(50)
source('exercise5.r')
```

We keep the genes of which we have the length(17907 of 17934) and we divide the transcription values of each gene by the gene length in order to eliminate the effect of the gene length.

```
load('dati3')
lunghezze<-read.delim('raw_trascr_count_annot.txt')
#get length
lunghezze<-lunghezze[,c('Symbol','Length')]
#keep genes for which we have the length
nomitr<-rownames(normali)
lunghezze<-lunghezze[lunghezze$Symbol%in%nomitr,]
lunghezze<-as.data.frame(lunghezze)
tenuti<-normali[lunghezze$Symbol,]
#divide by length
for (i in 1:length(tenuti[,1])){
  tenuti[i,]<-tenuti[i,]/
    (lunghezze[lunghezze$Symbol==rownames(tenuti)[i],])$Length
}
```

We now eliminate the 'flat' genes. To do so we calculate the coefficient of variation CV = (standard deviation) / (mean) and we keep the genes with CV > 0.3. We end up with 10898 genes.

```
deviazioni<-apply(tenuti,FUN=sd,1)
medie<-apply(tenuti,FUN=mean,1)
coeff<-deviazioni/medie
filtrati<-tenuti[coeff>0.3,]
```

We now scale the genes in order to make them comparable so that we can cluster them according to euclidean distance.

```
trasp<-t(filtrati)
trasp<-scale(trasp)
scalati<-t(trasp)
```

Visualize first rows of the obtained dataframe

```
print(scalati[1:3,])
```

```
##         Group2_S1 Group2_S2  Group2_S3  Group2_S5 Group2_S6   Group2_S7
## RFC2  -0.18630999 0.4706172 -0.7483490  0.7076488 -1.728307 -1.13339668
## HSPA6  1.88046815 0.3033801  0.4331318  0.6586117 -1.109339 -0.05940562
## PAX8   0.08448569 0.1676800  1.3715431 -0.6217536  3.464594  0.09848426
##        Group2_S8  Group2_S9 Group2_S10 Group2_S11 Group2_S12 Group2_S13
## RFC2  -1.4754152  0.2164129 -0.4543797 -0.7652092  0.6897199 -0.1793212
## HSPA6 -0.5447894 -0.5693958  1.8730672  1.3929547  0.4637057  0.1443584
## PAX8   0.4710075 -0.1432720 -0.6979540  0.7409776 -0.5707112  0.4549151
##       Group2_S14   Group2_S16   Group1_S1  Group1_S2  Group1_S3  Group1_S4
## RFC2   1.4638802  0.002959127 -0.04793178 -2.0783560 -0.6105302  0.2395498
## HSPA6 -0.5042990 -0.724046420  0.09330649 -0.9317607  2.5563381 -0.3643502
## PAX8  -0.3830795  1.154040096 -0.28697734  0.4763723  0.3819565 -0.8151737
##        Group1_S5      CTRL1      CTRL2      CTRL3      CTRL4       CTRL5
## RFC2   0.3586045  2.0820094  0.7234632  1.0170993  0.3690310  0.53055250
## HSPA6  0.4122141 -0.3020657 -0.9907012 -1.0657119 -0.9469663 -0.39521883
## PAX8  -0.3108892 -0.8238376 -1.2335868 -0.7419069 -1.7348790 -0.07052292
##           CTRL6      CTRL7
## RFC2  -0.8298208  1.3657791
## HSPA6 -0.7999683 -0.9035184
## PAX8   0.1836542 -0.6151668
```

Clustering

We do our analysis using all the subjects available in order to have as many instances of each gene as possible. All our models will be 'instantaneous' since our samples correspond to different subjects, not different time samples.

First of all we cluster our data for two main reasons: 1) Cluster that are very similar won't be distinguishable in terms of relations with other genes 2) We need to reduce the dimensionality of our data in order to be able to do the analysis in terms of computational cost.

The number of clusters we decide to identify is 4000. We saw that above this number the computational time becomes too big for our computers. Still the last clusters joined are fairly similar.

We use hierarchical clustering with average distance.

```
# computing distance matrix
dist_matrix <- dist(scalati, method = 'euclidean')
# average linkage is used to update distance matrix
hierarchical <- hclust(dist_matrix, method = 'average')
#we keep 4000 cluster
groups<-cutree(hierarchical,4000)
```

```
gruppi<-list()
for(i in 1:4000){
  gruppi<-append(gruppi,list(names(groups[groups==i])))
}
```

The variable gruppi will allow us the recover the genes belonging to each cluster starting from the cluster number. Let's visualize some of the clusters.

```
print(gruppi[1:3])
```

```
## [[1]]
## [1] "RFC2"  "POLD1"
##
## [[2]]
## [1] "HSPA6"  "SH3BP5" "ITGAM"
##
## [[3]]
##  [1] "PAX8"    "ASPG"    "TBX5"    "SOX9"    "ASPA"    "GABBR2"  "VAX2"
##  [8] "IGF2-AS" "BVES"    "PLIN4"
```

We now create the final dataframe with clusters on rows, subjects on columns and for each cluster the mean of the values of the genes belonging to the cluster.

```
tabella<-matrix(rep(0,104000),nrow=4000)
for (i in 1:4000){
  if (length(gruppi[[i]])==1){
    tabella[i,]<-scalati[gruppi[[i]],]
  }
  else{
    tabella[i,]<-colMeans(scalati[gruppi[[i]],])
  }
}
allsubjects<-tabella
colnames(allsubjects)<-colnames(scalati)
rownames(allsubjects)<-1:4000
```

Let's visualize first rows of the clusters dataframe.

```
print(allsubjects[1:3,])
```

```
##    Group2_S1  Group2_S2  Group2_S3  Group2_S5  Group2_S6  Group2_S7   Group2_S8
## 1  0.4671865  0.6192296 -0.3201538  0.7041883 -1.7656351 -1.2418749 -1.08532705
## 2  1.7625278  0.6030635 -0.2688751  0.8654969 -0.4633767  0.7156886 -0.38611794
## 3 -0.5648914 -0.5731483  0.1519936 -0.2577882  3.4769037  0.4879343 -0.03158122
##     Group2_S9 Group2_S10 Group2_S11 Group2_S12 Group2_S13 Group2_S14
## 1  0.09915618 -0.4576452  -1.202795  0.8477745 -0.5114286  1.0562849
## 2 -0.47153769  2.2022348   1.180014  0.2163313  0.3799675 -0.7438183
## 3  0.21839909 -0.3654366   1.004433 -0.8466321  0.2024500 -0.3948933
##    Group2_S16 Group1_S1  Group1_S2   Group1_S3  Group1_S4  Group1_S5      CTRL1
## 1 -0.08780388 0.1587399 -1.9135469 -0.07019445  0.1588446  0.9026023  1.8738125
## 2 -0.57922626 0.4454892 -0.7177345  1.60881234 -0.1420618 -0.4152900 -0.7141207
```

```
## 3  0.35011744 0.5687204  0.5535703 -0.14358722 -0.7716088 -0.3691097  0.1043765
##        CTRL2      CTRL3      CTRL4      CTRL5      CTRL6      CTRL7
## 1  0.9123835  0.6034425 -0.2974612  0.3234190 -0.8734038  1.1002051
## 2 -0.9549198 -0.8890480 -1.0670395 -0.5751840 -0.6646227 -0.9266528
## 3 -0.8576790 -0.2888835 -0.8729590 -0.3758294  0.2640608 -0.6689309
```

Partial correlation

The first method we use for inferring relations is partial correlation, assuming an instantaneous model.

First of all we calculate the partial correlation between clusters of genes.

```
dati<-allsubjects
dati<-as.matrix(dati)
partial_matrix <- pcor(t(dati), method = 'pearson')$estimate
rownames()
```

Let's visualize part of the matrix

```
rownames(partial_matrix)<-1:4000
colnames(partial_matrix)<-1:4000
print(partial_matrix[1:6,1:6])
```

```
##            1           2           3           4          5           6
## 1 1.00000000  0.09891935  0.37386470  0.05980658 0.12935270  0.39995603
## 2 0.09891935  1.00000000  0.08630857 -0.05830387 0.04312444 -0.16456850
## 3 0.37386470  0.08630857  1.00000000 -0.38137346 0.03475923 -0.27453347
## 4 0.05980658 -0.05830387 -0.38137346  1.00000000 0.14321936 -0.17093791
## 5 0.12935270  0.04312444  0.03475923  0.14321936 1.00000000  0.06224729
## 6 0.39995603 -0.16456850 -0.27453347 -0.17093791 0.06224729  1.00000000
```

To calculate the null hypothesis we create a matrix permuting the columns of allsubjects independently for each row. We then calculate the partial correlation of this new matrix. Our original idea was to create more than one permutation matrix, but it was computationally too heavy.

```
set.seed(50)
nulla<-dati
N<-dim(nulla)[1]
M<-dim(nulla)[2]
for (i in 1:N){
  nulla[i,]<-sample(dati[i,],M)
}
parnull<- pcor(t(nulla), method = 'pearson')$estimate
```

To decide the threshold our original idea was to estimate G0 and then look for the threshold that gave the desired FDR(ideally 0.05). Again, this was computationally too heavy, so we decided to be conservative assuming G0=G and we found the threshold empirically. We decided to set it to 0.8, this was the value that gave us the best balance between number of selected and FDR value.

```
soglia<-0.80
valori<-as.numeric(parnull)
valori<-valori[!(valori==1)]
```

```
vettorepar<-as.numeric(partial_matrix)
vettorepar<-vettorepar[!(vettorepar==1)]
totali<-length(valori)
totpar<-length(vettorepar)
selezionati<-length(vettorepar[abs(vettorepar)>=soglia])
expFP<-(length(valori[abs(valori)>=soglia])/totali)*totpar
FDR<-expFP/selezionati
risultato<-c(selezionati,expFP,FDR)
names(risultato)<-c('selected','expFP','FDR')
print(risultato)
```

```
##    selected       expFP         FDR
## 476.0000000 162.0000000   0.3403361
```

Still the FDR is fairly high, so we have to remember this when considering the obtained relatioins

We find the relations inferred with this threshold taking also the sign of the relation.

```
bool<-abs(partial_matrix)>soglia
parrelazioni<-which(bool==TRUE,arr.ind = TRUE)
parrelazioni<-as.data.frame(parrelazioni)
parrelazioni<-parrelazioni[!(parrelazioni[,1]==parrelazioni[,2]),]
segni<-c()
M<-dim(parrelazioni)[1]
for (i in 1:M){
  segni<-append(segni,sign(partial_matrix[parrelazioni[i,1],parrelazioni[i,2]]))
}
for (j in (1 :length(segni))){
  if (segni[j]==1){
    segni[j]<-'+'
  }
  else {
    segni[j]<-'-'
  }
}
parrelazioni<-cbind(parrelazioni,segni)
rownames(parrelazioni)<-1:M
colnames(parrelazioni)<-c('cluster1','cluster2','sign')
```

Let's visualize first relations

```
print(parrelazioni[,1:5])
```

```
##           1   2   3    4    5
## cluster1 44 311 815 1615 3223
## cluster2  4   5   5    5    5
## segno     -   -   -    -    -
```

Mutual information and ARACNe

The second method we use is the one that considers mutual information as a similraty between variables. We will use the correction introduced in ARACNe to keep just the relation that are effectively present.

First of all we calculate the mutual information for our data.

```
N <- dim(allsubjects)[1]
M <- dim(allsubjects)[2]
discretized <- allsubjects
discretized<-t(discretized)
discretized<-discretize(discretized,nbins=7,disc='equalwidth')
rownames(discretized)<-colnames(allsubjects)
mutualm<-matrix(rep(0,N^2),nrow=N)
rownames(mutualm)<-colnames(discretized)
colnames(mutualm)<-colnames(discretized)
mutualm<-mutinformation(discretized)
mutue<-mutualm
```

Let's visualize part of the matrix.

```
print(mutue[1:6,1:6])
```

```
##           1         2         3         4         5         6
## 1 1.8497035 0.5036590 0.7063723 0.2942254 0.5853518 0.6125325
## 2 0.5036590 1.6585261 0.4189637 0.1601449 0.3207305 0.6879502
## 3 0.7063723 0.4189637 1.2090307 0.2438397 0.2645931 0.4849250
## 4 0.2942254 0.1601449 0.2438397 0.6929082 0.1287594 0.2652405
## 5 0.5853518 0.3207305 0.2645931 0.1287594 1.4534989 0.4698538
## 6 0.6125325 0.6879502 0.4849250 0.2652405 0.4698538 1.7875246
```

To calculate the null hypothesis matrix we repeat the same procedure we did before but this time calculating the mutual information. As before, we could do just one permutation because of computational cost.

```
set.seed(50)
nulla<-allsubjects
N<-dim(nulla)[1]
M<-dim(nulla)[2]
for (i in 1:N){
  nulla[i,]<-sample(allsubjects[i,],M)
}
nulla<-t(nulla)
nulla<-discretize(nulla,nbins=7,disc='equalwidth')
rownames(nulla)<-colnames(allsubjects)
nullmut<-matrix(rep(0,N^2),nrow=N)
nullmut<-mutinformation(nulla)
```

We calculate the entropy of both cases, we will use it to scale the mutual information to eliminate the dependency on entropy.

Firstly we do that for our original data.

```
dati<-allsubjects
N=dim(dati)[1]
dati<-t(dati)
dati<-as.data.frame(dati)
dati<-infotheo::discretize(dati,nbins=7,disc='equalwidth')
names_df <- colnames(dati)
```

```
values_df <- rep(0, N)
entropy_df <- data.frame(entropia = values_df, row.names = names_df)
for (i in (1:N)){
  row_curr <- dati[,i]
  entropy_df[i,] <- entropy(row_curr)
}
print(head(entropy_df))
```

```
##    entropia
## 1 1.8497035
## 2 1.6585261
## 3 1.2090307
## 4 0.6929082
## 5 1.4534989
## 6 1.7875246
```

Then for the permuted data(using our function).

```
nullentropy<-entropia(nulla)
```

We know scale each mutual information value dividing it by the max of the two entropies

Firstly we do that for our original data

```
scalmut<-mutue
N<-dim(scalmut)[1]
for (i in 1:N){
  for (j in 1:N){
    scalmut[i,j]<-scalmut[i,j]/max(entropy_df[i,],entropy_df[j,])
  }
}
```

Let's visualize part of the matrix

```
print(scalmut[1:6,1:6])
```

```
##           1          2         3          4          5         6
## 1 1.0000000 0.27229176 0.3818841 0.15906626 0.31645711 0.3311517
## 2 0.2722918 1.00000000 0.2526120 0.09655858 0.19338283 0.3848620
## 3 0.3818841 0.25261204 1.0000000 0.20168195 0.18203875 0.2712830
## 4 0.1590663 0.09655858 0.2016819 1.00000000 0.08858585 0.1483843
## 5 0.3164571 0.19338283 0.1820387 0.08858585 1.00000000 0.2628517
## 6 0.3311517 0.38486195 0.2712830 0.14838427 0.26285169 1.0000000
```

Then for the permute data(using our function)

```
scalnulmut<-scalamutua(nullmut,nullentropy)
```

As before, for the same reasons, we find the threshold empirically looking for a good balance between the number of selected genes and the FDR.

```
soglia=0.63
selezionati<-length(which(scalmut>=soglia&scalmut!=1))
valori<-length(which(scalnulmut>=soglia&scalnulmut!=1))
alfa<-valori/length(which(scalnulmut!=1))
expFP<-alfa*length(which(scalmut!=1))
FDR<-expFP/selezionati
risultato<-c(selezionati,expFP,FDR)
names(risultato)<-c('selected','expFP','FDR')
print(risultato)
```

```
##    selected       expFP         FDR
## 156.0000000  44.0000000   0.2820513
```

We select relations with mutual information greater than the threshold and visualize the first of them

```
maggiori<-which(scalmut>soglia &scalmut!=1, arr.ind = TRUE)
print(head(maggiori))
```

```
##       row col
## 1554 1554   6
## 136   136  13
## 2353 2353  17
## 311   311  23
## 134   134  33
## 2648 2648  33
```

Now we check if there are relations to eliminate according to the information theory theorem used in ARACNe. This theorem states that if there exists a relation between variables X and Y, and a relation between variables X and Z, but not between Y and Z, then MI(Y,Z)<max{MI(X,Y),MI(X,Z)}.

```
unici<-unique((maggiori[,1]))
triplette<-c()
for (i in unici){
  opposite<-maggiori[maggiori[,1]==i,2]
  for (j in opposite){
    opposite2<-maggiori[maggiori[,1]==j,2]
    for (k in opposite2){
      if (k %in% opposite){
        triplette<-append(triplette,c(i,j,k))

      }
    }
  }
}
print(triplette)
```

```
## NULL
```

Triplette is empty, so there are no relations to eliminate.

Mutual information and entropy

The third method we use is the one that searches for couples of genes x,y for which H(x)=M(x,y); in this case we can also infer the direction of the relation y->x .

We already have the mutual information matrix saved in mutue and entropy in entropy_df from the previous analysis.

```
relazioni<-matrix(c(0,0),nrow=1)
for (i in (1:N)){
  for (j in (1:N)){
    if (i!=j){
      if (entropy_df[i,]==mutue[i,j]){
        relazioni<-rbind(relazioni,c((colnames(allsubjects))[i],
                                     (rownames(allsubjects))[j]))
      }
    }
  }
}
relazioni<-relazioni[-1,]
colnames(relazioni)<-c('causa','effetto')
```

We visualize the first relations obtained.

```
print(relazioni[,1:5])
```

```
##           1   2   3   4    5
## causa    16  16  16  16   16
## effetto 156 293 852 985 1948
```

```
library(bnlearn)
```

Bayesian Network

The last method we used is the bayesian network.

The algorithm we used for optimizing the network is Restricted Maximization (rsmax2 in R). One of the problems of bayesian networks is the hugeness of the search space and most of the time is spent in searching for regulators of a variable that are improbable. This algorithm instead sppeds up the computation considering only a maximum number k of candidates for a variable at each iteration and maximazes the score according to these restrictions.

```
allsubjects <- as.data.frame(allsubjects)
rete_bayes <- rsmax2(allsubjects)
archi <- rete_bayes$arcs
```

We visualize the first relations obtained

```
print(archi[,1:5])
```

```
##        1   2   3    4    5
## from 183  44  44 1675  513
## to   340 213 209 1712 1014
```

Comparing Relations

We want to see if there are common relations between the ones found with different methods. These may be key relations.

The four sets of relations are: -relazioni -> the ones found with M(x,y)=H(x) -parrelazioni -> the ones found through partial correlation -archi -> the ones found with the bayesian network -relaracne -> the ones found with M(x,y) as similarity

```
intersezione(relazioni,parrelazioni)
```

```
## data frame con 0 colonne e 0 righe
```

```
intersezione(relazioni,archi)
```

```
##          168 211 453  403 437  164
## cluster1  16 142 142  142 264  264
## cluster2 852 629 867 3081 491 2346
```

```
intersezione(relazioni,relaracne)
```

```
## data frame con 0 colonne e 0 righe
```

```
intersezione(parrelazioni,archi)
```

```
## data frame con 0 colonne e 0 righe
```

```
intersezione(parrelazioni,relaracne)
```

```
## data frame con 0 colonne e 0 righe
```

```
intersezione(relaracne,archi)
```

```
##           16  97    5  355   15
## cluster1  33  13  513 1195 1712
## cluster2 134 136 1014 1563 1690
```

Genetic Relations

Boolean network

For the second part we focus on genetic data. The idea is to see if there are relations between different SNPs, in the sense that maybe some mutated SNP occur frequently when specific others are present or when others are missing.

To do this we modelled each SNP as a boolean variable that has value 1 if it is mutated, 0 if it is the reference one.

First of all we transform genetic data into boolean form.

```
genetici <- read.delim('GeneticData.txt', row.names = 1)
N <- dim(genetici)[1]
M <- dim(genetici)[2]
row_names <- rownames(genetici)
col_names <- colnames(genetici)
booleana_tab <- matrix(rep(0, N*M), nrow = N)
rownames(booleana_tab) <- row_names
colnames(booleana_tab) <- col_names
for (i in (1:N)){
  for (j in (1:M)){
    if (genetici[i,j] == '---'){
      booleana_tab[i,j] <- 0
    }
    else{
      booleana_tab[i,j] <- 1
    }
  }
}
print(booleana_tab[1:6,1:6])
```

```
##               Group1_S1 Group1_S2 Group1_S3 Group1_S4 Group1_S5 Group2_S1
## chr1_11169676         0         0         0         0         0         1
## chr1_11174331         0         0         0         0         0         1
## chr1_11181327         0         1         1         1         1         1
## chr1_11181418         0         0         0         0         0         0
## chr1_11181457         0         0         0         0         0         1
## chr1_11181985         0         0         0         0         0         0
```

We eliminate rows of all 0's or of all 1's. We go from 3061 SNPs to 2973.

```
N <- dim(booleana_tab)[1]
M <- dim(booleana_tab)[2]
lista_indici <- c()
for (i in (1:N)){
  somma <- sum(booleana_tab[i,])
  if ((somma < 1) | (somma == M)){
    lista_indici <- append(lista_indici, i)
  }
}
booleana_tab <- booleana_tab[-lista_indici,]
```

Clustering equal rows. Height = 0 means no differences, so relative rows are equal; 'rimanenti' contains rows that are all different, so by cutting the clustering tree at rimanenti + 1 we obtain what we want

```
dist_matrix <- dist(booleana_tab, method = 'euclidean')
cluster_boo <- hclust(dist_matrix, method = 'average')
altezze <- cluster_boo$height
rimanenti <- length(altezze[altezze > 0])
rimanenti <- rimanenti + 1
clustering <- cutree(cluster_boo, rimanenti)
gruppi_boo<-list()
for(i in 1:rimanenti){
```

```
  gruppi_boo<-append(gruppi_boo,list(names(clustering[clustering==i])))
}
```

We create a table with all clusters; 714 is the number of obtained clusters, starting from a total of 2973 genes.

```
tabella_boo<-matrix(rep(0,14994),nrow=714)
for (i in 1:714){1
  if (length(gruppi_boo[[i]])==1){
    tabella_boo[i,]<-booleana_tab[gruppi_boo[[i]],]
  }
  else{
    righe <- booleana_tab[gruppi_boo[[i]],]
    riga <- righe[1,]
    tabella_boo[i,]<-riga
  }
}
colnames(tabella_boo)<-colnames(booleana_tab)
rownames(tabella_boo)<-1:714
```

We use the reveal algorithm to look for possible relations between the clusters of SNP's. We tried to implement the algorithm with k=2(usign pairs of regulators), but it was computationally too heavy, so we just looked for relations of the type MI(x,y) = H(x).

First of all we computed the entropy for all clusters

```
N=dim(tabella_boo)[1]
tabella_boo<-t(tabella_boo)
names_df <- colnames(tabella_boo)
values_df <- rep(0, N)
entropy_df <- data.frame(entropia = values_df, row.names = names_df)
for (i in (1:N)){
  row_curr <- tabella_boo[,i]
  entropy_df[i,] <- entropy(row_curr)
}
print(head(entropy_df))
```

```
##    entropia
## 1 0.1914441
## 2 0.6829081
## 3 0.1914441
## 4 0.1914441
## 5 0.1914441
## 6 0.3144922
```

Then we computed the mutual information

```
tabella_boo <- as.data.frame(tabella_boo)
discreti_bool <- infotheo::discretize(tabella_boo, nbins = 2, disc = 'equalwidth')
mutual_bool<-mutinformation(discreti_bool)
print(mutual_bool[1:6, 1:6])
```

```
##             1          2           3           4           5           6
## 1 0.191444082 0.02753781 0.002381945 0.002381945 0.002381945 0.125430065
## 2 0.027537806 0.68290810 0.027537806 0.041944612 0.027537806 0.057028653
## 3 0.002381945 0.02753781 0.191444082 0.002381945 0.002381945 0.004889369
## 4 0.002381945 0.04194461 0.002381945 0.191444082 0.002381945 0.004889369
## 5 0.002381945 0.02753781 0.002381945 0.002381945 0.191444082 0.125430065
## 6 0.125430065 0.05702865 0.004889369 0.004889369 0.125430065 0.314492201
```

And finally we find the relations between clusters We visualize the first obtained

```
relazioni_bool<-matrix(c(0,0),nrow=1)
for (i in (1:N)){
  for (j in (1:N)){
    if (i!=j){
      if (entropy_df[i,]==mutual_bool[i,j]){
        relazioni_bool<-rbind(relazioni_bool,c((rownames(mutual_bool))[i],
                                               (rownames(mutual_bool))[j]))
      }
    }
  }
}
relazioni_bool<-relazioni_bool[-1,]
colnames(relazioni_bool)<-c('causa','effetto')

print(head(relazioni_bool))
```

```
##      causa effetto
## [1,] "1"   "138"
## [2,] "4"   "489"
## [3,] "5"   "8"
## [4,] "6"   "93"
## [5,] "8"   "5"
## [6,] "9"   "328"
```

Genetic affecting transcriptomics

Here in the third part we search if there are some SNP's that significantly affect the transcription of the corresponding gene

Data preparation

First of all we associate to each SNP the corresponding gene

```
dati<-read.delim('GeneticData.txt',row.names=1)
annotazione<-read.delim('ensembl_annot.txt')
cromosomi<-row.names(dati)
simboli<-annotazione$Symbol
simboli<-annotazione$Variation.ID
associazione<-matrix(nrow=length(cromosomi))
colnames(associazione)<-'Gene'
geni<-c()
```

```
for (i in cromosomi){
  geni<-append(geni,annotazione$Symbol[grep(unlist(strsplit(i,'_'))[2],simboli)[1]])
}
associazione[,1]<-geni
rownames(associazione)<-rownames(dati)
associazione<-na.omit(associazione)
```

We visualize some of the associations

```
print(associazione[25:35,])
```

```
##  chr1_11307894  chr1_11307899  chr1_11307944  chr1_11307950  chr1_11308010
##          "MTOR"         "MTOR"         "MTOR"         "MTOR"         "MTOR"
##  chr1_11308020  chr1_11308037  chr1_11308070 chr1_117057386 chr1_117057470
##          "MTOR"         "MTOR"         "MTOR"         "CD58"         "CD58"
## chr1_117057488
##          "CD58"
```

Then we keep just the SNPs whose associated gene is present in the transcription data and we keep just the genes for which there is a SNP associated to them

```
unici<-na.omit(unique(geni))
raw<-read.delim('raw_trascr_count.txt',row.names=1)
totali<-rownames(raw)
comuni<-unici[unici%in%totali]
dati<-cbind(dati,geni)
comgenetic<-dati[dati$geni%in%comuni,]
raw<-raw[rownames(raw)%in%comuni,]
raw<-raw[,1:19]
raw<-cbind(raw,rownames(raw))
colnames(raw)[length(colnames(raw))]<-'geni'
soggettigen<-colnames(comgenetic)
soggettiraw<-colnames(raw)
comgenetic<-comgenetic[,c(soggettigen%in%soggettiraw)]
```

We make the genetic data a boolean network (1 mutation, 0 no mutation)

```
ultima<-comgenetic[,length(comgenetic[1,])]
comgenetic<-booleana(comgenetic, file=FALSE)
comgenetic[,length(comgenetic[1,])]<-ultima
comgenetic<-as.data.frame(comgenetic)
print(head(comgenetic),3)
```

```
##               Group1_S1 Group1_S2 Group1_S3 Group1_S4 Group1_S5 Group2_S1
## chr1_11169676         0         0         0         0         0         1
## chr1_11174331         0         0         0         0         0         1
## chr1_11181327         0         1         1         1         1         1
## chr1_11181418         0         0         0         0         0         0
## chr1_11181457         0         0         0         0         0         1
## chr1_11181985         0         0         0         0         0         0
##               Group2_S2 Group2_S3 Group2_S5 Group2_S6 Group2_S7 Group2_S8
```

```
## chr1_11169676         0         0         0         0         0         0
## chr1_11174331         0         0         0         0         0         0
## chr1_11181327         1         0         0         1         1         0
## chr1_11181418         0         0         0         0         0         0
## chr1_11181457         0         0         0         0         0         0
## chr1_11181985         0         0         0         0         0         1
##               Group2_S9 Group2_S10 Group2_S11 Group2_S12 Group2_S13 Group2_S14
## chr1_11169676         0          0          0          0          0          0
## chr1_11174331         0          0          0          0          0          0
## chr1_11181327         1          1          1          0          0          0
## chr1_11181418         0          0          1          0          0          0
## chr1_11181457         0          0          0          0          0          0
## chr1_11181985         0          0          0          0          0          0
##               Group2_S16 geni
## chr1_11169676          0 MTOR
## chr1_11174331          0 MTOR
## chr1_11181327          0 MTOR
## chr1_11181418          0 MTOR
## chr1_11181457          0 MTOR
## chr1_11181985          0 MTOR
```

Correlation

We now search for high correlation values between mutation presence or absence and corresponding gene trascription

We create the correlation matrix after having ordered the subjects in the same way

```
tabcor<-matrix(rep(0,length(rownames(comgenetic))),
               nrow=length(rownames(comgenetic)))
rownames(tabcor)<-rownames(comgenetic)
comgenetic<-comgenetic[,order(colnames(comgenetic))]
raw<-raw[,order(colnames(raw))]
for (i in 1:length(rownames(comgenetic))){
  gene<-comgenetic$geni[i]
  tabcor[i]<-cor(as.numeric(comgenetic[i,2:length(comgenetic[1,])])
                 ,as.numeric(raw[gene,2:length(raw[1,])]))
}
```

We select the SNPs for which the correlation is higher than 0.7

```
posizioni<-which(abs(tabcor)>=0.7)
tenute<-tabcor[posizioni,]
```

And we plot the SNP values with the transcription values of the corresponding graph for these selected SNPs. We save the plot on the file 'snpimportanti.pdf'

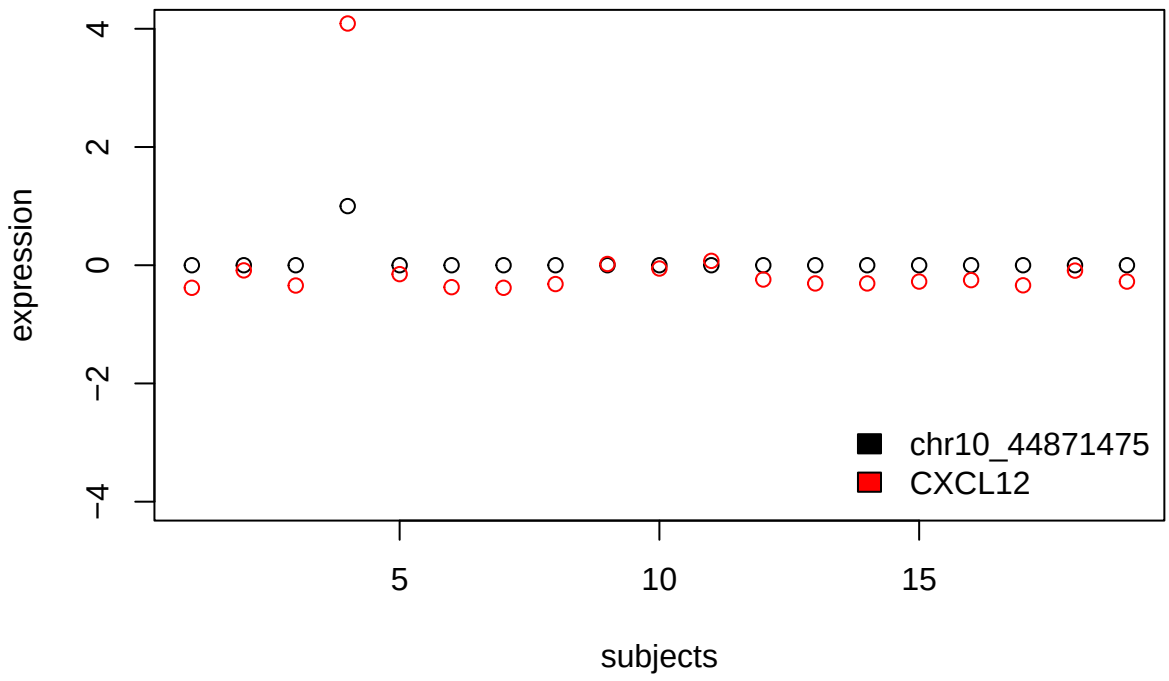
```
pdf('snpimportanti.pdf')
par(mfrow=c(2,2))
for (j in 1:length(tenute)){
  snp<-names(tenute)[j]
  gene<-comgenetic[snp,]$geni
  plot(1:19,comgenetic[snp,2:20], ylim=c(-4,4),xlab='subjects',ylab='expression')
```

```
  vettore<-raw[gene,2:20]
  vettore<-t(vettore)
  vettore<-scale(vettore)
  vettore<-t(vettore)
  points(1:19,vettore,col='red')
  legend(x='bottomright',c(snp,gene), fill=c('black','red'),bty='n')
}
dev.off()
```

```
## pdf
##   2
```

We can look at one of this plots to see that it's informative

```
snp<-names(tenute)[1]
gene<-comgenetic[snp,]$geni
plot(1:19,comgenetic[snp,2:20], ylim=c(-4,4),xlab='subjects',ylab='expression')
vettore<-raw[gene,2:20]
vettore<-t(vettore)
vettore<-scale(vettore)
vettore<-t(vettore)
points(1:19,vettore,col='red')
legend(x='bottomright',c(snp,gene), fill=c('black','red'),bty='n')
```



The presence of the mutated SNP causes a great increase in the value of the transcription.