Investigating the coherency between logical relationships and gene expressions for each connected gene pair in the signaling networks

Mehran Piran

December 12, 2018

piranmehran@gmail.com
Researchgate

Introduction

Study the correlation between the gene expression profiles at both mRNA and protein level is common. Several studies have shown that mRNA and protein expression are poorly correlated. There are some reasons such as miRNA activity on mRNA transcripts or post-translational modifications which cause the variation in amounts of mRNAs and active/inactive Proteins. However, the association between mRNA and protein of housekeeping genes which is not mostly related to signaling processes has been also reported. Now, another major question is the association between mRNA or protein level with the logical relations inside the wiring diagram of the signaling networks. Considering the sign of interactions between components, does the gene expression or protein amount help the signal to be transduced or not? In two simple elements, there are eight possible states (figure 1). Four of these states conform to the direction of the signal transduction, i.e., if the interaction is activation, both of two related components are up-regulated or down-regulated; and if the interaction is inhibitory, the expression of two associated components is inverse (Panel A). The other four possible states are against to signal transduction's direction (panel B). Analyzing these two different clusters of states in various biological conditions could be an interesting subject for the discussion. Figure 2 is the flowchart which depicts the different steps to reach the data needed to analyze the coherency between the gene pairs in KEGG signaling pathways.

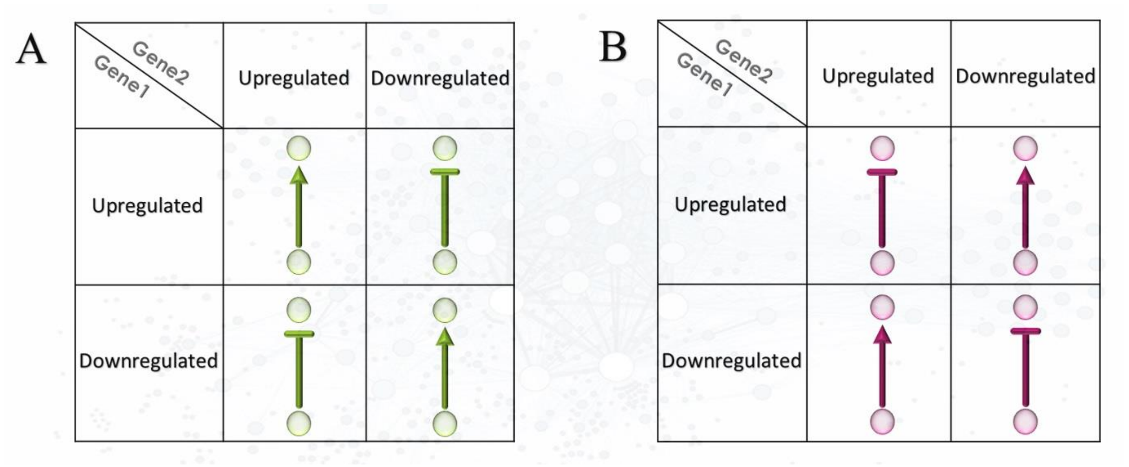


Figure1

The Procedure

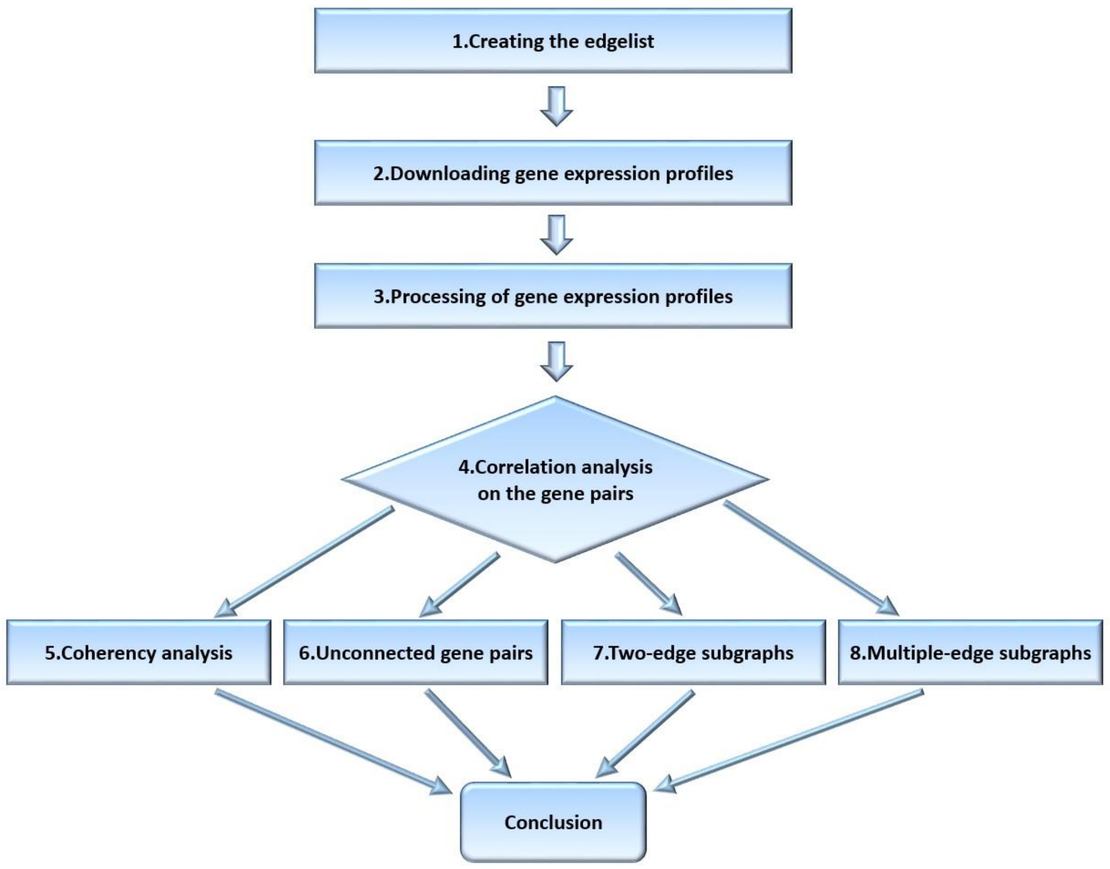

1.Creating the edgelist
2.Downloading gene expression profiles
3.Processing of gene expression profiles
4.Correlation analysis on the gene pairs
5.Coherency analysis
6.Unconnected gene pairs
7.Two-edge subgraphs
8.Multiple-edge subgraphs
Conclusion

Figure2: Different steps to analyze the coherency between the gene pairs

1.Creating an edgelist from KEGG database

An edgelist was created from all human KEGG pathways using "KEGGgraph" package.

1.1.Downloading KEGG pathways

All the human signaling pathways were downloaded in form of KGML files from KEGG database KEGG Pathways. There were 206 human pathways on July 24th, 2017.

An example of these pathways is: Ras Signaling Pathway

1.2.Impoting pathways into R in form of graphs

Downloaded pathways were imported into R using the combination of “parseKGML2Graph” and “lapply” functions. The “parseKGML2Graph” is a function which creates a class of graph (graphNEL) from each pathway and is suitable for networks with few edges and high nodes. All 206 graphs were stored in KGMLGraphs object.

```
library(KEGGgraph)

#Define path
kgfiles<-list.files("F:/Projects/Project Coherency/KEGG GEO/kegg
xml",full.names=T)

#Read kgml files
KGMLGraphs=lapply(kgfiles,parseKGML2Graph,genesOnly=F)
length(KGMLGraphs)

## [1] 206

head(KGMLGraphs)

## [[1]]
## A graphNEL graph with directed edges
## Number of Nodes = 94
## Number of Edges = 247
##
## [[2]]
## A graphNEL graph with directed edges
## Number of Nodes = 115
## Number of Edges = 311
##
## [[3]]
## A graphNEL graph with directed edges
## Number of Nodes = 64
## Number of Edges = 18
##
## [[4]]
## A graphNEL graph with directed edges
## Number of Nodes = 84
## Number of Edges = 79
##
## [[5]]
## A graphNEL graph with directed edges
## Number of Nodes = 93
## Number of Edges = 282
##
## [[6]]
## A graphNEL graph with directed edges
## Number of Nodes = 267
## Number of Edges = 856
```

1.3.Extracting edge information

Edge information is derived from each pathway using "getKEGGedgeData" function, and they are stored in **l** object.

```
l = lapply(KGMLGraphs,getKEGGedgeData)
length(l)

## [1] 206
```

1.4. Constructing the edgelists

The following codes construct the edgelists from all 206 signaling pathways and remove the non-informative information. The loop contains two "if" conditions to exclude edges without complete data. Note that, nodes (e.g., hsa: 1950) are genes and Subtype is the relationship between genes, e.g., activation, inhibition and so on. A large list called **listedgelists** is created, and each element contains an edgelist.

```
# an example of complete informative edge:
l[[1]][[1]]

##   KEGG Edge (Type: PPrel):
## ------------------------------------------------------------
## [ Entry 1 ID ]: hsa:1950
## [ Entry 2 ID ]: hsa:1956
## [ Subtype ]:
##   [ Subtype name ]: activation
##   [ Subtype value ]: -->
## ------------------------------------------------------------

# First exclusive condition: There is no edge between nodes. For example:
KGMLGraphs[[26]]

## A graphNEL graph with directed edges
## Number of Nodes = 133
## Number of Edges = 0

# Second exclusive condition: There is no information about the edge. For
example:
l[[10]][[251]]

##   KEGG Edge (Type: PCrel):
## ------------------------------------------------------------
## [ Entry 1 ID ]: hsa:6543
## [ Entry 2 ID ]: cpd:C01330
## [ Subtype ]:
## ------------------------------------------------------------
```

```
a = NULL
class(a)

## [1] "NULL"

b = list()

listedgelists = list()

for(i in 1:length(l)){
  if(is.list(l[[i]]) & length(l[[i]]) == 0){
    listedgelists[[i]] = c(0,0,0)
  } else
  {
  d = matrix(0,length(l[[i]]),3)
  d = as.data.frame(d)
  for(j in 1:length(l[[i]])){
    d[j,1] = l[[i]][[j]]@entry1ID
    d[j,2] = l[[i]][[j]]@entry2ID
    c1 = l[[i]][[j]]@subtype$subtype

    if(class(c1) == class(a)){
      d[j,3] =  "none"
    } else
    {
      d[j,3] = l[[i]][[j]]@subtype$subtype@name
    }

  }

    listedgelists[[i]] = d
  }
}
```

listedgelists object contains all 206 edgelists.

1.5.Merging all edgelists

All the edgelists are attached together and a large edgelist is created.

```
table = do.call("rbind", listedgelists)
dim(table)

## [1] 68757     3
```

Duplicated rows are omitted as below.

```
table1 = table[!duplicated(table),]
```

1.6. Preprocessing the edgelist

In the following code, “activation” and “inhibition” interactions are separated from each other.

```
idx1 = which(table1[,3] == "activation")

idx2 = which(table1[,3] == "inhibition")

idx = c(idx1,idx2)

table2 = table1[idx,]

dim(table2)

## [1] 28870     3
```

After that, only edges which are KEGG Ids starting with “hsa” are selected.

```
idx3 = grep("^hsa" , table2[,1])

idx4 = grep("^hsa" , table2[,2])

idx5 = intersect(idx3,idx4)

table3 = table2[idx5,]

dim(table3)

## [1] 26490     3
```

```
sum(!grepl("^hsa" , table3[,1]))

## [1] 0

sum(!grepl("^hsa" , table3[,2]))

## [1] 0
```

Next, all the KEGG Ids are changed into gene Ids.

```
kgid1 = as.character(table3[,1])

kgid2 = as.character(table3[,2])

geneid1 <- translateKEGGID2GeneID(kgid1)

geneid2 <- translateKEGGID2GeneID(kgid2)

any(is.na(geneid1))

## [1] FALSE

any(is.na(geneid1))

## [1] FALSE
```

Gene Ids are mapped to gene symbols using "org.Hs.eg.db" package which contains annotation for the human genome.

```
require(org.Hs.eg.db)

genesymbol1 <- sapply(mget(geneid1, org.Hs.egSYMBOL,
                            ifnotfound=NA), "[[",1)
length(genesymbol1)

## [1] 26490

any(is.na(genesymbol1))

## [1] TRUE

genesymbol2 <- sapply(mget(geneid2, org.Hs.egSYMBOL,
                           ifnotfound=NA), "[[",1)
length(genesymbol2)

## [1] 26490

any(is.na(genesymbol2))

## [1] TRUE
```

1.7. Final edgelist

So far, there are three columns in the edgelist. The first two columns contain interacting genes and the last one shows interaction type: activation or inhibition

```
edgelist = table3
```

```r
edgelist[,1] = genesymbol1

edgelist[,2] = genesymbol2

dim(edgelist)

## [1] 26490     3
```

Finally, an ID column is added to the edgelist.

```r
id1 = paste0("E0000" , 1:9 )
id2 = paste0("E000" , 10:99)
id3 = paste0("E00" , 100:999)
id4 = paste0("E0" , 1000:9999)
id5 = paste0("E" , 10000:length(edgelist[,1]))
ID = c(id1,id2,id3,id4,id5)
edgelist = cbind(ID,edgelist)
colnames(edgelist) = c("ID" , "Gene1" , "Gene2" , "Interaction type")
edgelist = apply(edgelist , 2 , as.character)
edgelist = as.data.frame(edgelist , stringsAsFactors = F)
class(edgelist)

## [1] "data.frame"
```

```r
head(edgelist)

##       ID Gene1 Gene2 Interaction type
## 1 E00001   EGF  EGFR       activation
## 2 E00002  TGFA  EGFR       activation
## 3 E00003   HGF   MET       activation
## 4 E00004   MET ERBB3       activation
## 5 E00005  IGF1 IGF1R       activation
## 6 E00006 VEGFA   KDR       activation
```

2.Downloading all up or down gene expression profiles

All the genes are stored at **edgelistGenes** object.

```r
edgelistGenes = unique(c(edgelist[,2],edgelist[,3]))

length(edgelistGenes)
```

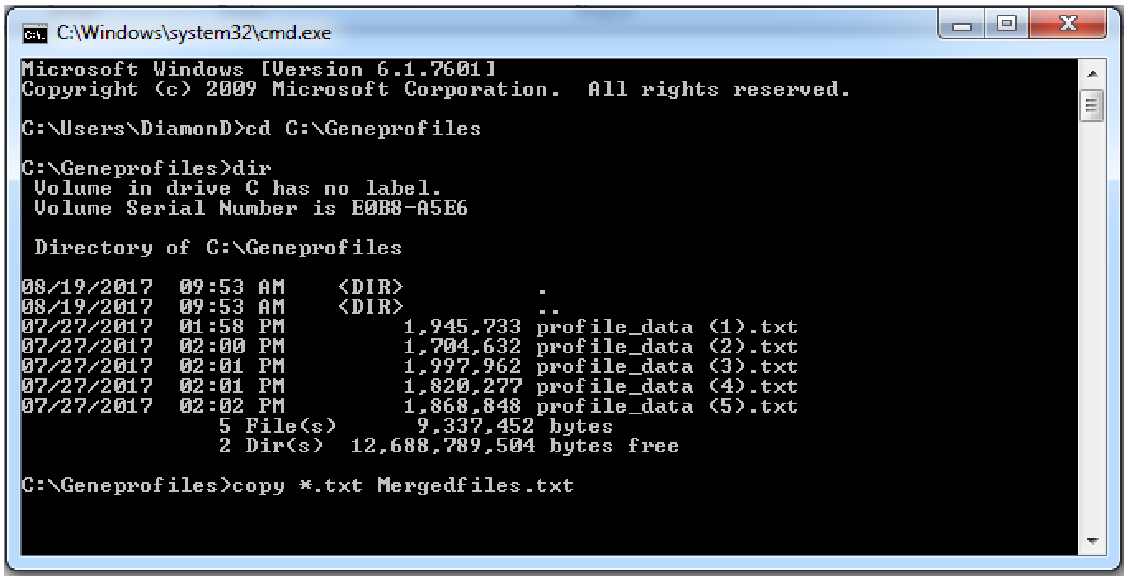
```
## [1] 3187
```

There were 3187 unique genes in the KEGG edgelist. We downloaded all the human up-down gene expression profiles for these genes from GEO database (https://www.ncbi.nlm.nih.gov/geoprofiles/). This database contains gene expression profiles derived from curated GEO DataSets and presents them as a chart that displays the expression level of one gene across all Samples within a DataSet. Due to the limitation in NCBI query rules (importing less than 200 gene names in the query and downloading 500 gene profiles in each step), we downloaded 423 text files, each contains nearly 500 gene profiles. A query example for some of the genes is shown here: GEO Profile.

A query example:
("Homo sapiens"[Organism] OR Homo sapiens[All Fields]) AND ("PPIA"[Gene Symbol] OR"GRB2"[Gene Symbol] OR"RAC1"[Gene Symbol] OR"AP2B1"[Gene Symbol] OR"GNAI1"[Gene Symbol] OR"PPP3R1"[Gene Symbol] OR"YWHAZ"[Gene Symbol] AND "up down genes"[filter])

2.1.Merging all gene expression profiles into one file

All the downloaded gene profiles are merged into Geneprofiles.txt as below (figure3):
1) all the text files are placed in a folder in drive C.
2) In the Windows7 command line shell, the following codes were run.
. cd C: Geneprofiles (Enter)
. dir (Enter) (Shows all the files in the directory)
. copy *.txt Geneprofiles.txt (Enter)

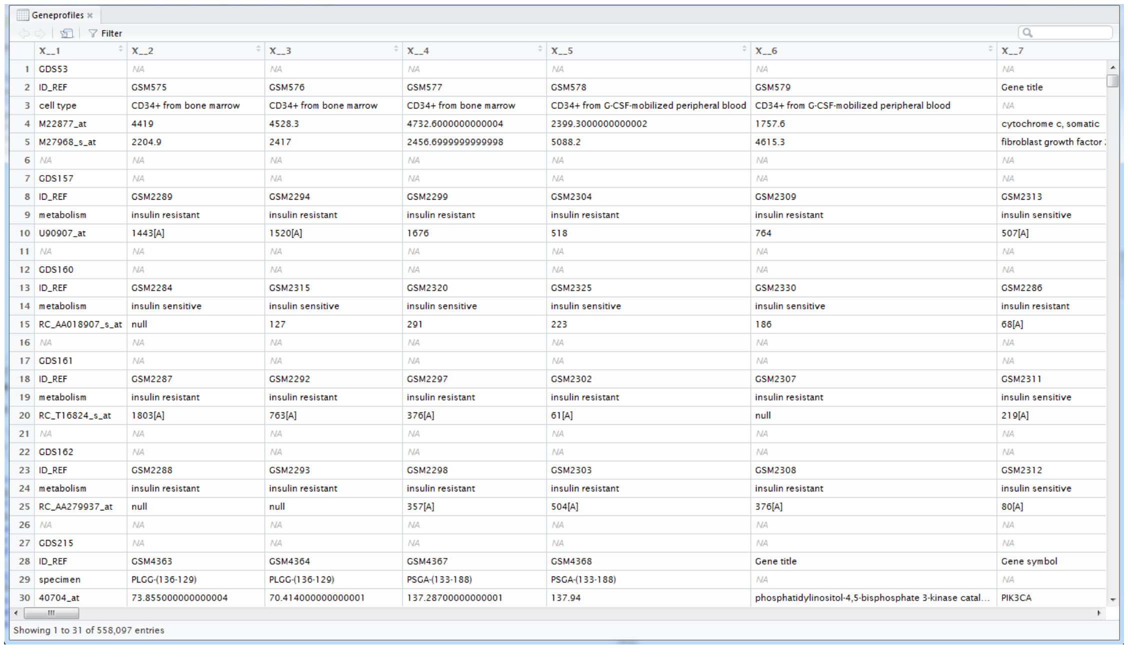


```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DiamonD>cd C:\Geneprofiles

C:\Geneprofiles>dir
 Volume in drive C has no label.
 Volume Serial Number is E0B8-A5E6

 Directory of C:\Geneprofiles

08/19/2017  09:53 AM    <DIR>          .
08/19/2017  09:53 AM    <DIR>          ..
07/27/2017  01:58 PM         1,945,733 profile_data (1).txt
07/27/2017  02:00 PM         1,704,632 profile_data (2).txt
07/27/2017  02:01 PM         1,997,962 profile_data (3).txt
07/27/2017  02:01 PM         1,820,277 profile_data (4).txt
07/27/2017  02:02 PM         1,868,848 profile_data (5).txt
               5 File(s)      9,337,452 bytes
               2 Dir(s)  12,688,789,504 bytes free

C:\Geneprofiles>copy *.txt Mergedfiles.txt
```

Figure3

2.2.Importing Geneprofiles file into R

```r
library(readxl)
Geneprofiles = read_excel("Geneprofiles.xlsx", col_names = F)
dim(Geneprofiles)
Geneprofiles = as.data.frame(Geneprofiles)
```



	X__1	X__2	X__3	X__4	X__5	X__6	X__7
1	GDS53	NA	NA	NA	NA	NA	NA
2	ID_REF	GSM575	GSM576	GSM577	GSM578	GSM579	Gene title
3	cell type	CD34+ from bone marrow	CD34+ from bone marrow	CD34+ from bone marrow	CD34+ from G-CSF-mobilized peripheral blood	CD34+ from G-CSF-mobilized peripheral blood	NA
4	M22877_at	4419	4528.3	4732.6000000000004	2399.3000000000002	1757.6	cytochrome c, somatic
5	M27968_s_at	2204.9	2417	2456.6999999999998	5088.2	4615.3	fibroblast growth factor :
6	NA	NA	NA	NA	NA	NA	NA
7	GDS157	NA	NA	NA	NA	NA	NA
8	ID_REF	GSM2289	GSM2294	GSM2299	GSM2304	GSM2309	GSM2313
9	metabolism	insulin resistant	insulin resistant	insulin resistant	insulin resistant	insulin resistant	insulin sensitive
10	U90907_at	1443[A]	1520[A]	1676	518	764	507[A]
11	NA	NA	NA	NA	NA	NA	NA
12	GDS160	NA	NA	NA	NA	NA	NA
13	ID_REF	GSM2284	GSM2315	GSM2320	GSM2325	GSM2330	GSM2286
14	metabolism	insulin sensitive	insulin sensitive	insulin sensitive	insulin sensitive	insulin sensitive	insulin resistant
15	RC_AA018907_s_at	null	127	291	223	186	68[A]
16	NA	NA	NA	NA	NA	NA	NA
17	GDS161	NA	NA	NA	NA	NA	NA
18	ID_REF	GSM2287	GSM2292	GSM2297	GSM2302	GSM2307	GSM2311
19	metabolism	insulin resistant	insulin resistant	insulin resistant	insulin resistant	insulin resistant	insulin sensitive
20	RC_T16824_s_at	1803[A]	763[A]	376[A]	61[A]	null	219[A]
21	NA	NA	NA	NA	NA	NA	NA
22	GDS162	NA	NA	NA	NA	NA	NA
23	ID_REF	GSM2288	GSM2293	GSM2298	GSM2303	GSM2308	GSM2312
24	metabolism	insulin resistant	insulin resistant	insulin resistant	insulin resistant	insulin resistant	insulin sensitive
25	RC_AA279937_at	null	null	357[A]	504[A]	376[A]	80[A]
26	NA	NA	NA	NA	NA	NA	NA
27	GDS215	NA	NA	NA	NA	NA	NA
28	ID_REF	GSM4363	GSM4364	GSM4367	GSM4368	Gene title	Gene symbol
29	specimen	PLGG-(136-129)	PLGG-(136-129)	PSGA-(133-188)	PSGA-(133-188)	NA	NA
30	40704_at	73.855000000000004	70.414000000000001	137.28700000000001	137.94	phosphatidylinositol-4,5-bisphosphate 3-kinase catal…	PIK3CA

Showing 1 to 31 of 558,097 entries

3.Processing gene expression profiles

3.1.Creating a large list called listGeneprofiles which contains each part of Geneprofiles dataframe in one of its elements

The following codes are to create a large list called **listGeneprofiles**. each element of the list contains one row for the GDS name, one row for ID_REF, GSMs, Gene ids and Gene symbol, and multiple rows for the expression values.

An empty list called listGeneprofiles is created. To extract the information of each GDS part, rows indexes containing “GDS” are found. A “loop” is used to exert some operations on each GDS part. v1 is a vector which contains the row after GDS i (the row which start with ID_REF). “index” is a vector holding all the elements of v1 which start with GSM letters. “l” shows how many GSMs are present in row idx[i] + 1. “d” is a data frame as the same dimention as each GDS part. The number of rows in matrix d, is equal to The number of rows starting from GDS[i] to GDS[i+1] minus two (“length((idx[i]):(idx[i+1] - 2))”). The number of columns in object d, is equal to the sum of column one as well as all the GSM columns and two columns after GSM columns (length(c(1,index,index[l]+c(1,2))))). Whenever d is completed for GDS i, extra rows are deleted. In other words, row one and rows which contain expression values (have “at” or numbers at the end) are kept.

```r
listGeneprofiles = list()
idx = grep("^GDS" , Geneprofiles[,1])
for(i in 1:(length(idx)-1)){

  v1 =  data[idx[i]+1,]
  index = grep("^GSM" , v1)
  l = length(index)

  d = as.data.frame(matrix(0,length((idx[i]):(idx[i+1] - 2)) ,
length(c(1,index,index[l]+c(1,2)))))
  for(j in 1:length((idx[i]):(idx[i+1] - 2))) {
    d[j,] = Geneprofiles[idx[i]+j-1,c(1,index,index[l]+c(2,3))]
  }

  colnames(d) = d[2,]
  idxx = grep(paste(paste(c(0:9,'_at'),"$",sep=""),collapse="|"),d[,1])
  listGeneprofiles[[i]] = d[idxx,]

  #print(i)
}

length(listGeneprofiles)

## [1] 73460
```

listGeneprofiles contains 73460 elements. each element has a similar structure as below.

```
listGeneprofiles[[1]]

##        ID_REF GSM575 GSM576             GSM577             GSM578 GSM579
## 1       GDS53   <NA>   <NA>               <NA>               <NA>   <NA>
## 4   M22877_at   4419 4528.3 4732.6000000000004 2399.3000000000002 1757.6
## 5 M27968_s_at 2204.9   2417 2456.6999999999998             5088.2 4615.3
##   Gene symbol Gene ID
## 1        <NA>    <NA>
## 4        CYCS   54205
## 5        FGF2    2247

listGeneprofiles[[57777]]

##        ID_REF           GSM136055          GSM136056          GSM136057
## 1     GDS2494                <NA>               <NA>               <NA>
## 5 200065_s_at             3630.35             3685.9            3662.81
## 6 202512_s_at 48.658799999999999 49.640799999999999              50.71
## 7 221497_x_at 162.16499999999999            128.107 138.99700000000001
##            GSM136064          GSM136065          GSM136066
GSM136058
## 1               <NA>               <NA>               <NA>
<NA>
## 5            3622.76            3423.19            3399.93
3083.94
## 6 72.458799999999997 55.823999999999998 71.829800000000006
38.535200000000003
## 7 90.802700000000002 82.813800000000001 78.484999999999999
142.18299999999999
##            GSM136059          GSM136060    Gene symbol         Gene ID
## 1               <NA>               <NA>           <NA>            <NA>
## 5            3125.62            3177.01 MIR3620///ARF1 100500810///375
## 6 38.714700000000001 43.994900000000001           ATG5            9474
## 7            156.648 143.34800000000001          EGLN1           54583
```

3.2.Creating a list called preListGDS which contains each GDS (dataset) in one of its elements

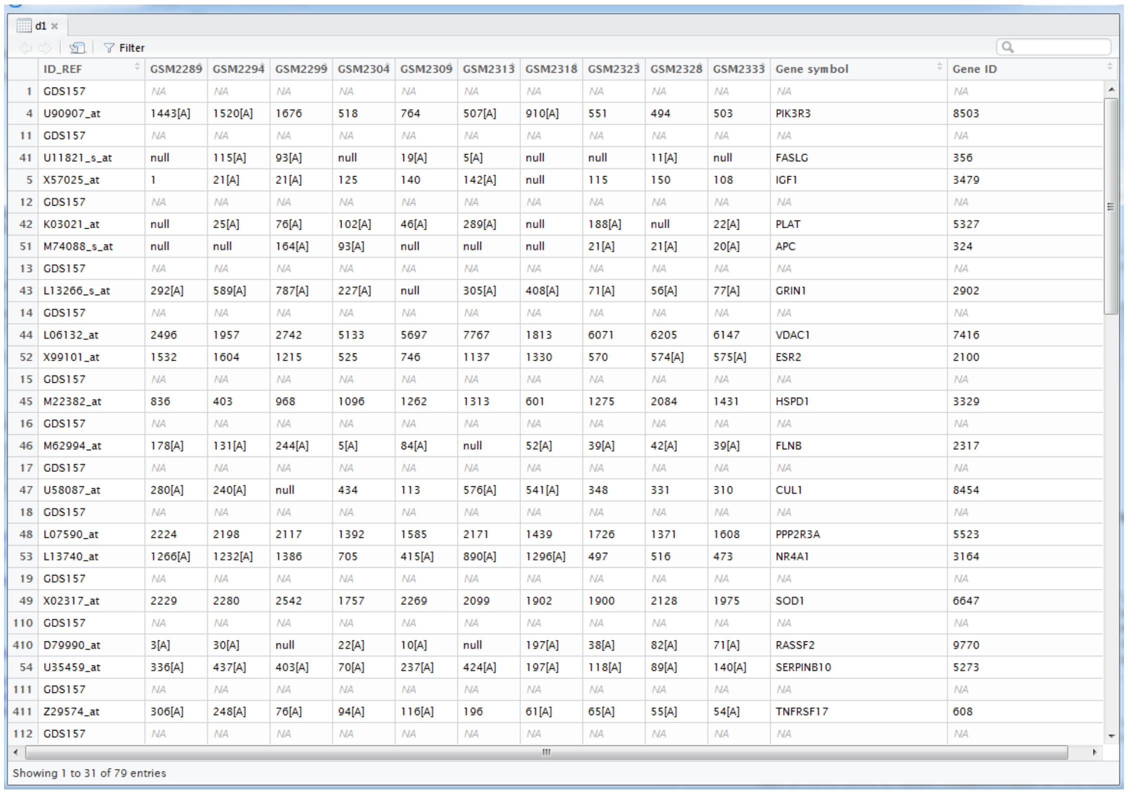
The following codes show how many GDSes (datasets) does exist in the “listGeneprofiles” object.

```
gds = c()
for(i in 1:length(listGeneprofiles)){
  gds[i] = listGeneprofiles[[i]][1,1]
```

```r
}
gds1 = unique(gds)

length(gds)

## [1] 73460
```

Same datasets are merged as below.

```r
preListGDS = list()
for(i in 1:length(gds)){
  idx = which(gds1[i]==gds)
  preListGDS[[i]] = do.call("rbind",listGeneprofiles[idx])
}
length(preListGDS)

## [1] 1693

# There are 1693 datasets
```



d1 ×

Filter

	ID_REF	GSM2289	GSM2294	GSM2299	GSM2304	GSM2309	GSM2313	GSM2318	GSM2323	GSM2328	GSM2333	Gene symbol	Gene ID
1	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	U90907_at	1443[A]	1520[A]	1676	518	764	507[A]	910[A]	551	494	503	PIK3R3	8503
11	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
41	U11821_s_at	null	115[A]	93[A]	null	19[A]	5[A]	null	null	11[A]	null	FASLG	356
5	X57025_at	1	21[A]	21[A]	125	140	142[A]	null	115	150	108	IGF1	3479
12	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
42	K03021_at	null	25[A]	76[A]	102[A]	46[A]	289[A]	null	188[A]	null	22[A]	PLAT	5327
51	M74088_s_at	null	null	164[A]	93[A]	null	null	null	21[A]	21[A]	20[A]	APC	324
13	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
43	L13266_s_at	292[A]	589[A]	787[A]	227[A]	null	305[A]	408[A]	71[A]	56[A]	77[A]	GRIN1	2902
14	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
44	L06132_at	2496	1957	2742	5133	5697	7767	1813	6071	6205	6147	VDAC1	7416
52	X99101_at	1532	1604	1215	525	746	1137	1330	570	574[A]	575[A]	ESR2	2100
15	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
45	M22382_at	836	403	968	1096	1262	1313	601	1275	2084	1431	HSPD1	3329
16	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
46	M62994_at	178[A]	131[A]	244[A]	5[A]	84[A]	null	52[A]	39[A]	42[A]	39[A]	FLNB	2317
17	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
47	U58087_at	280[A]	240[A]	null	434	113	576[A]	541[A]	348	331	310	CUL1	8454
18	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
48	L07590_at	2224	2198	2117	1392	1585	2171	1439	1726	1371	1608	PPP2R3A	5523
53	L13740_at	1266[A]	1232[A]	1386	705	415[A]	890[A]	1296[A]	497	516	473	NR4A1	3164
19	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
49	X02317_at	2229	2280	2542	1757	2269	2099	1902	1900	2128	1975	SOD1	6647
110	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
410	D79990_at	3[A]	30[A]	null	22[A]	10[A]	null	197[A]	38[A]	82[A]	71[A]	RASSF2	9770
54	U35459_at	336[A]	437[A]	403[A]	70[A]	237[A]	424[A]	197[A]	118[A]	89[A]	140[A]	SERPINB10	5273
111	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
411	Z29574_at	306[A]	248[A]	76[A]	94[A]	116[A]	196	61[A]	65[A]	55[A]	54[A]	TNFRSF17	608
112	GDS157	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Showing 1 to 31 of 79 entries

Figure5

Some datasets does not contain gene symbols or gene ids. We delete them as below.

```
v = c()
for(i in 1:length(preListGDS)){
  v[i]="Gene symbol" %in% colnames(preListGDS[[i]]) & "Gene ID" %in%
colnames(preListGDS[[i]])
}

index = which(!v)
index

## [1]  933 1062 1229 1332

l = preListGDS[index]

c(l[[1]][1,1],l[[2]][1,1],l[[3]][1,1],l[[4]][1,1])

## [1] "GDS2771" "GDS3795" "GDS3268" "GDS4206"

colnames(l[[2]])

##   [1] "ID_REF"    "GSM483301" "GSM483302" "GSM483303" "GSM483305"
"GSM483307"
##   [7] "GSM483312" "GSM483313" "GSM483317" "GSM483318" "GSM483319"
"GSM483322"
##  [13] "GSM483327" "GSM483328" "GSM483330" "GSM483332" "GSM483333"
"GSM483336"
##  [19] "GSM483337" "GSM483339" "GSM483351" "GSM483352" "GSM483354"
"GSM483358"
##  [25] "GSM483384" "GSM483386" "GSM483388" "GSM483390" "GSM483391"
"GSM483396"
##  [31] "GSM483399" "GSM483400" "GSM483401" "GSM483412" "GSM483418"
"GSM483420"
##  [37] "GSM483421" "GSM483426" "GSM483428" "GSM483431" "GSM483436"
"GSM483442"
##  [43] "GSM483443" "GSM483444" "GSM483447" "GSM483448" "GSM483450"
"GSM483455"
##  [49] "GSM483458" "GSM483461" "GSM483462" "GSM483464" "GSM483466"
"GSM483468"
##  [55] "GSM483476" "GSM483477" "GSM483300" "GSM483308" "GSM483310"
"GSM483311"
##  [61] "GSM483323" "GSM483338" "GSM483353" "GSM483361" "GSM483363"
"GSM483364"
##  [67] "GSM483366" "GSM483368" "GSM483371" "GSM483372" "GSM483373"
"GSM483374"
##  [73] "GSM483379" "GSM483380" "GSM483381" "GSM483389" "GSM483404"
"GSM483405"
##  [79] "GSM483410" "GSM483411" "GSM483413" "GSM483416" "GSM483417"
"GSM483419"
##  [85] "GSM483427" "GSM483433" "GSM483434" "GSM483445" "GSM483459"
"GSM483465"
```

```
##  [91] "GSM483470" "GSM483473" "GSM483478" "GSM483304" "GSM483315"
"GSM483320"
##  [97] "GSM483325" "GSM483329" "GSM483331" "GSM483334" "GSM483341"
"GSM483343"
## [103] "GSM483344" "GSM483347" "GSM483348" "GSM483349" "GSM483350"
"GSM483356"
## [109] "GSM483362" "GSM483365" "GSM483367" "GSM483369" "GSM483370"
"GSM483375"
## [115] "GSM483376" "GSM483377" "GSM483378" "GSM483385" "GSM483402"
"GSM483403"
## [121] "GSM483406" "GSM483407" "GSM483408" "GSM483414" "GSM483415"
"GSM483424"
## [127] "GSM483437" "GSM483439" "GSM483440" "GSM483446" "GSM483449"
"GSM483454"
## [133] "GSM483456" "GSM483460" "GSM483463" "GSM483471" "GSM483297"
"GSM483298"
## [139] "GSM483299" "GSM483306" "GSM483309" "GSM483314" "GSM483316"
"GSM483321"
## [145] "GSM483324" "GSM483326" "GSM483335" "GSM483340" "GSM483342"
"GSM483345"
## [151] "GSM483346" "GSM483355" "GSM483357" "GSM483359" "GSM483360"
"GSM483382"
## [157] "GSM483383" "GSM483387" "GSM483392" "GSM483393" "GSM483394"
"GSM483395"
## [163] "GSM483397" "GSM483398" "GSM483409" "GSM483422" "GSM483423"
"GSM483425"
## [169] "GSM483429" "GSM483430" "GSM483432" "GSM483435" "GSM483438"
"GSM483441"
## [175] "GSM483451" "GSM483452" "GSM483453" "GSM483457" "GSM483467"
"GSM483469"
## [181] "GSM483472" "GSM483474" "GSM483475" "GSM483479" "GSM483480"
"GSM483481"
## [187] "GSM483482" "GSM483483" "GSM483484" "GSM483485" "GSM483486"
"GSM483487"
## [193] "GSM483488" "GSM483489" "0"         "0"
```

```
preListGDS = preListGDS[-index]
length(preListGDS)

## [1] 1689
```

GDS rows are deleted from each element of preListGDS as below.

```
for(i in 1:length(preListGDS)){
  preListGDS[[i]] = preListGDS[[i]][!grepl("^GDS",preListGDS[[i]][,1]),-1]
}
```

3.3.Editing preListGDS object into ListGDS

Datasets in **preListGDS** are required to be edited; so, a “for loop” along with 4 conditions are created to edit preListGDS. edited preListGDS is called **ListGDS**. In the gene symbol column of most of GDSes, there are multiple gene symbols for one value. In that case, the gene symbol which is in the edgelist is selected.

```r
ListGDS = preListGDS

# The first condition determines if there is a need for editing

# The second condition is for when there are multiple gene symbols for one
value and
# more than one of them is mapped to the edgelist genes.

# Thr third condition is for when there are multiple gene symbols for one
value,
# but none of them mapped to the gene set "Gene"

# The forth condition is for when there are multiple gene symbols for one
value and
# only one of them is mapped to the gene set "Gene""


for(i in 1:length(ListGDS)){

  l1 = length(ListGDS[[i]][1,]) - 1

  a = ListGDS[[i]][,l1]

  a = strsplit(a,"///",fixed = T)

  l = unlist(lapply(a,length))

  idxx = which(l>1)

    if(length(idxx>0)){
    a1 = a[idxx]
    for(j in 1:length(a1)){
      idxx1 = which(a1[[j]] %in% edgelistGenes)
      if(length(idxx1>1)){
        ListGDS[[i]][,l1][idxx[j]] = a[idxx][[j]][idxx1[1]]
      } else if(length(idxx1)==0){
        ListGDS[[i]][,l1][idxx[j]] = a[idxx][[j]][1]
      } else {
        ListGDS[[i]][,l1][idxx[j]] = a[idxx][[j]][idxx1]
      }
    }
  }

  # Deleting regular [A] characters next to the some values
  # changing class of values to numeric
  # changing order of columns

  ListGDS[[i]] = cbind(ListGDS[[i]][,l1,drop=F],ListGDS[[i]][,1:(l1-
1),drop=F])
  ListGDS[[i]] = as.data.frame(ListGDS[[i]],stringsAsFactors=F)

  for(j in 2:l1){
    ListGDS[[i]][,j] = sub("\\[A]","",ListGDS[[i]][,j])
    ListGDS[[i]][,j] = as.numeric(ListGDS[[i]][,j])
  }

   #print(i)
}
```

```
head(ListGDS[[1]] , 20)
```

```
##     Gene symbol  GSM575  GSM576  GSM577  GSM578  GSM579
## 4          CYCS  4419.0  4528.3  4732.6  2399.3  1757.6
## 5          FGF2  2204.9  2417.0  2456.7  5088.2  4615.3
## 41         GAS6   765.9   562.4   667.5   364.8   395.6
## 51         GRB2   226.2   212.6   198.2    10.3    34.3
## 42        IL2RB   367.8   508.4   328.2    57.9    34.3
## 52           F3   573.5   607.5   356.9    79.9    21.6
## 6        CASP10  1765.1  1400.3  1843.8   660.6   479.1
## 7        PTPRM   676.4   622.7   611.0   309.4   298.9
## 43       IQGAP1   508.0   482.1   480.6   323.9   328.9
## 44       SREBF1  8303.8  9411.2  8260.5  3804.7  4592.9
## 53         FGD1  2637.9  2035.4  1447.3  5435.8  4638.0
## 61          RHO  6781.8  6509.0  7001.6 10833.1 11141.8
## 45         THRA 21144.2 45869.4 52175.2  9517.2  6752.6
## 46       PLA2R1   721.7   375.6   462.5  5271.3  6598.9
## 47        RAB7A  4117.5  4042.4  2926.3 10662.4 11609.5
## 48         SDC2  1846.2  1794.0  2030.2   656.4   776.9
## 54           C5   167.4    41.9    33.5   326.9   295.8
## 49      PPP2R5A  6597.4  7377.2  8328.5  3951.5  3474.7
## 410        IRF9   445.3   315.8   283.7  2369.0  3024.7
## 411    SERPINB10   570.1   478.5   465.0   123.3   109.5
```

```
head(preListGDS[[1]] , 20)
```

```
##                  GSM575               GSM576             GSM577
GSM578
## 4                  4419               4528.3 4732.6000000000004
2399.3000000000002
## 5                2204.9                 2417 2456.6999999999998
```

```
5088.2
## 41                765.9              562.4              667.5
364.8
## 51             226.2[A]           212.6[A]           198.2[A]
10.3[A]
## 42                367.8              508.4              328.2
57.9
## 52                573.5              607.5              356.9
79.9[A]
## 6                1765.1             1400.3             1843.8
660.6[A]
## 7                 676.4 622.70000000000005                611
309.4[A]
## 43               508[A]           482.1[A]           480.6[A]
323.9[A]
## 44  8303.7999999999993 9411.2000000000007             8260.5
3804.7
## 53               2637.9             2035.4             1447.3
5435.8
## 61               6781.8               6509             7001.6
10833.1
## 45              21144.2            45869.4 52175.199999999997
9517.2000000000007
## 46                721.7              375.6              462.5
5271.3
## 47               4117.5             4042.4             2926.3
10662.4
## 48               1846.2               1794             2030.2
656.4
## 54             167.4[A]            41.9[A]            33.5[A]
326.89999999999998
## 49               6597.4             7377.2             8328.5
3951.5
## 410               445.3              315.8              283.7
2369
## 411               570.1              478.5                465
123.3[A]
##                  GSM579 Gene symbol Gene ID
## 4                1757.6        CYCS   54205
## 5                4615.3        FGF2    2247
## 41                395.6        GAS6    2621
## 51              34.3[A]        GRB2    2885
## 42              34.3[A]       IL2RB    3560
## 52                 21.6          F3    2152
## 6              479.1[A]      CASP10     843
## 7   298.89999999999998       PTPRM    5797
## 43                328.9      IQGAP1    8826
## 44  4592.8999999999996      SREBF1    6720
## 53                 4638        FGD1    2245
## 61              11141.8         RHO    6010
## 45              6752.6        THRA    7067
## 46              6598.9      PLA2R1   22925
## 47             11609.5       RAB7A    7879
## 48               776.9        SDC2    6383
## 54            295.8[A]          C5     727
## 49              3474.7     PPP2R5A    5525
## 410             3024.7        IRF9   10379
## 411              109.5   SERPINB10    5273
```

```
class(ListGDS[[1]][,4])
```

```
## [1] "numeric"
```

```
# We keep gene symbols and delete columns related to gene Ids

dim(ListGDS[[1]])
```

```
## [1] 49  6
```

```
dim(preListGDS[[1]])
```

```
## [1] 49  7
```

The following code, shows the indices in GDSes containing repetitious GSMs.

```
indexx = list()
for(i in 1:length(ListGDS)){

  logic=c()

  for(j in (1:length(ListGDS))[-i]){
    logic[j]=any(colnames(ListGDS[[i]])[-1] %in% colnames(ListGDS[[j]]))
  }
  index = which(logic)
  if(length(index) > 0){
    indexx[i] = index
  }
  print(i)
}

index.of.GDSes.having.repetitious.GSMs = which(unlist(lapply(indexx ,
length)) > 0)
```

In the code below, repetitious GSMs are deleted in different GDSes.

```
for(i in 1:length(ListGDS)){

  logic=c()
```

```r
  for(j in (1:length(ListGDS))[-i]){
    logic[j]=any(colnames(ListGDS[[i]])[-1] %in% colnames(ListGDS[[j]]))
  }
  index = which(logic)
  if(length(index) > 0){
    for(z in 1:length(index)){
      logicc = colnames(ListGDS[[index[z]]]) %in% colnames(ListGDS[[i]])[-1]
      indexx = which(logicc)
      ListGDS[[index[z]]]=ListGDS[[index[z]]][,-indexx]
    }
  }
  # print(i)
}
```

Indices in ListGDS for which there is no GSM left are omitted.

```r
index = which(unlist(lapply(lapply(lapply(ListGDS , dim) , function(x) x[2])
, function(x) length(x) == 0)))
ListGDS = ListGDS[-index]

# All the datasets have more than two GSMs
which(unlist(lapply(lapply(lapply(ListGDS , dim) , function(x) x[2]),
function(x) x < 2)))

## integer(0)
```

Indices in ListGDS for which there is no gene left are removed.

```r
index = which(unlist(lapply(lapply(lapply(ListGDS , dim) , function(x) x[1])
, function(x) x < 2)))
ListGDS = ListGDS[-index]
```

All the gene names that are in the ListGDS object, are stored in an object called **profilesGenes**.

```r
profilesGenes = c()

for(i in 1:length(ListGDS)){
  profilesGenes = c(profilesGenes,ListGDS[[i]][,1])
}

profilesGenes = unique(profilesGenes)

any(is.na(profilesGenes))

profilesGenes = profilesGenes[-which(is.na(profilesGenes))]
```

```
length(profilesGenes)

## [1] 3207
```

The first row in the KEGG edgelist contains "EGF" gene. Check how many datasets contain this gene.

```
which(profilesGenes == "EGF")

## [1] 1679

logic=c()
for(i in 1:length(ListGDS)){
  logic[i] =  profilesGenes[1679] %in% ListGDS[[i]][,1]
}
idxng1679 = which(logic)
length(which(logic))

## [1] 44

head(ListGDS[[idxng1679[20]]],10)

##    Gene symbol GSM1200171 GSM1200172 GSM1200173 GSM1200174
## 4       PRKACB  3762.2700  3588.7100  5354.8300  5143.6300
## 5          FAS   204.5730   195.6560   507.2090   473.8230
## 6          FAS   178.3230   169.3150   401.8230   366.3830
## 7          EGF    72.6697    66.5497    22.1397    22.2993
## 8         CYCS  8361.6400  8924.8400  5821.4400  6205.0000
## 9          HGF    36.7687    40.6940    85.0087    88.0442
## 10         HGF    51.4297    47.0476    83.9657    83.3615
## 11         FAS   107.4120   103.9990   317.9370   290.5730
## 12         FAS    60.8529    55.7002   161.3550   140.7340
## 13        CYCS    40.3911    42.7323    20.5259    17.2582

length(profilesGenes) # 3207 is the number of genes extracted from all the
gene profiles

## [1] 3207

length(edgelistGenes) # 3187 is the number of genes at the edgelist

## [1] 3187
```

common genes between gene profiles and edgelist are stored in **Genes** object.

```
sum(profilesGenes %in% edgelistGenes)

Genes = profilesGenes[which(profilesGenes %in% edgelistGenes)]
```

```
length(Genes)

## [1] 3047

# There are 3047 common genes between the gene profiles and the edgelist
```

In the folowing code all the GSM names are stored at **gsms** object.

```
list3=list()
for(i in 1:(length(ListGDS))){
  index = grep("^GSM" , colnames(ListGDS[[i]]))
  list3[[i]] = colnames(ListGDS[[i]])[index]
}
length(list3)

gsm = unlist(list3)

gsms = unique(gsm)

length(gsms)

## [1] 40903
```

3.4.Constructing an expression matrix called Exprtable which contains all the gene values amongst the GSMs

First of all, an NA matrix called **Exprtable** is created which has genes in row and GSMs in column. for each gene name, a sublist is created by a loop. For gene one, we check which GDSes of ListGDS object contain it. Then, we store these GDSes at a **sublist** object. After that, for each element in the sublist, the rows indicating the gene i are stored at **idxx** object. If there is just one row, we put that row in Exprtable object. Otherwise, the row with larger IQR is located in Exprtable expression matrix.

```
Exprtable = matrix(NA,length(Genes),length(gsms))

Exprtable = as.data.frame(Exprtable)

rownames(Exprtable) = Genes

colnames(Exprtable) = gsms

dim(Exprtable)

## [1]  3047 40903
```

```
for(i in 1:length(Genes)){


  logic = c()
  for(j in 1:length(ListGDS)){
    logic[j] = Genes[i] %in% ListGDS[[j]][,1]
  }
  sublist = ListGDS[logic]


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

    gsm = colnames(sublist[[j]])[-1]

      idxx = which(sublist[[j]][,1] == Genes[i])

      if(length(idxx) == 1){
        Exprtable[Genes[i],gsm] = sublist[[j]][idxx,-1]
      } else {
           d = sublist[[j]][idxx,-1]
           IQRs = c()
          for(z in 1:length(d[,1])){
           a=as.numeric(na.omit(as.numeric(d[z,])))
           IQRs[z] = quantile(a, 0.75 ) - quantile(a, 0.25)
           Exprtable[Genes[i],gsm] = d[which.max(IQRs),]
         }

        }
      }
      #print(i)
    }

sum(is.na(Exprtable))

## [1] 120876590

sum(!(is.na(Exprtable)))

## [1] 3754851
```

3.5.designing a list called preSignalingNet1 which contains expression values for each edge

In this step, both **Exprtable** and **edglist** are used. For each edge in the edgelist, a data frame is created. First of all, an empty list called preSignalongNet is created. source gene of row i of the edgelist is stored at gene1 object. Target gene of the same row is stored at gene2 object. The source and target gene rows in Exprtable are stored at a and b objects respectively. If there exist more than two non-NA GSMs in both source and target genes, these values are saved as a dataframe in element i of preSignalingNet1 object. If not, we put “empty”" in preSignalingNet1[i].

```
preSignalingNet1 = list()
for(i in 1:length(edgelist[,1])){
  gene1 = edgelist[,2:3][i,1]
  gene2 = edgelist[,2:3][i,2]

  idx1 = which(gene1 == rownames(Exprtable))
  idx2 = which(gene2 == rownames(Exprtable))
  a = Exprtable[idx1,]
  b = Exprtable[idx2,]
  l1 = !is.na(a)
  l2 = !is.na(b)

  if(length(intersect(which(l1),which(l2))) > 0){
    d = rbind(a,b)
    idx = intersect(which(l1),which(l2))
    preSignalingNet1[[i]] = d[,idx]
    names(preSignalingNet1)[i] = edgelist[i,1]
  } else {
    preSignalingNet1[[i]] = "empty"
    names(preSignalingNet1)[i] = edgelist[i,1]
  }
  #print(i)
}
```

The information for the edge 10000 is diepicted as follow. THBS4 is the source gene and CD47 is the target gene.

```
edgelist[10000,]

##            ID Gene1 Gene2 Interaction type
## 10000 E10000 THBS4  CD47       activation

preSignalingNet1[[10000]]

##        GSM183695 GSM185526 GSM185527 GSM185528 GSM185529 GSM185530
GSM185531
## THBS4   86.5063   80.9516  13.80250   285.017  310.7550   39.8476
521.398
## CD47    20.8250   33.1526   3.27527    11.985   32.0414   24.6110
3.422
##        GSM185532 GSM185533 GSM185534 GSM185535 GSM185536 GSM47867 GSM47868
## THBS4 254.00900 686.64900 525.22700  404.1640 733.72700     33.9     36.0
## CD47    7.69714   4.67268   7.08067   29.6316   9.12542     90.9     26.7
##        GSM47700 GSM47862 GSM47869 GSM47863 GSM47864 GSM47870 GSM47871
GSM47865
```

```
## THBS4     23.2     31.1     22.2      6.3      6.1     91.4     58.3
61.7
## CD47      38.5     50.4     74.3     75.0     44.1      4.8     22.8
4.3
##       GSM47866 GSM112271 GSM112272 GSM112273 GSM112274 GSM112275 GSM112276
## THBS4     56.3   -0.7179   -0.1888   -0.7167   -0.6063   -0.9728   -0.8966
## CD47       1.7    0.4725    0.3915    0.0521    0.1217   -0.5224   -0.4888
##       GSM112277 GSM112278 GSM112279 GSM112280 GSM112281 GSM112282
GSM112283
## THBS4   -1.1997   -2.0078   -0.9423   -0.6307   -1.0251   -1.0449   -
1.2765
## CD47    -0.8899   -0.9997    0.0867    0.1363   -0.5541   -0.5307   -
1.0612
##       GSM112284 GSM112285 GSM112286 GSM112287 GSM112288 GSM112289
GSM112290
## THBS4   -1.3115    0.4004    0.2038    0.1565    0.3431   -0.5992   -
0.6471
## CD47    -1.1395    0.6933    0.6035    0.2097    0.2109    0.0369
0.0715
##       GSM112293 GSM112294 GSM112295 GSM112296 GSM112297 GSM112298
## THBS4    0.1637    0.1986   -0.8115   -0.6926   -0.8596   -0.8864
## CD47     0.4232    0.4921   -0.0582   -0.2459   -0.5638   -0.7999
```

The Object **idx.NO.gsms** has the indices of the edges having less than three GSMs.

```
idx.NO.gsms = which(preSignalingNet1 %in% "empty" )

length(idx.NO.gsms)

## [1] 10650
```

3.7.Creating preSignalingNet2

All the values in each preSignalingNet1 element are separated based on the source of datasets and they are stored at **preSignalingNet2** object. Furthermore, for some indices of preSignalingNet1, source and target genes are the same and they have exactly the same values. These indices are stored at **index.of.not.matching.preSignalingNet.and.ListGDS** object and they are assigned empty in preSignalingNet2 object. **'These indices are related to loops in the edgelist.'**

```
preSignalingNet2 = preSignalingNet1

index.of.not.matching.preSignalingNet.and.ListGDS=c()

for(i in (1:length(preSignalingNet1))[-idx.NO.gsms]){
  # Here the index object get indices of all the GDSes which have created
signalingnet[[i]] and have the source and target genes
  logic=c()
  for(j in 1:length(ListGDS)){
    logic[j]=any(colnames(preSignalingNet1[[i]]) %in% colnames(ListGDS[[j]]))
& all(rownames(preSignalingNet1[[i]]) %in% ListGDS[[j]][,1])
  }
  index = which(logic)

  if(length(index) > 1){
    coexpr = ListGDS[index]
    coEXpr = list()
    colNAMES = lapply(coexpr,colnames)
    for(z in 1:length(coexpr)){
      coEXpr[[z]]=preSignalingNet1[[i]][ , colNAMES[[z]][which( colNAMES[[z]]
%in% colnames(preSignalingNet1[[i]]))]]

    }
    preSignalingNet2[[i]] = coEXpr

  } else if(length(index) == 1){
    preSignalingNet2[[i]] = preSignalingNet1[[i]]

  } else {
    preSignalingNet2[[i]] = "empty"
    index.of.not.matching.preSignalingNet.and.ListGDS[i] = i
  }

}

index.of.not.matching.preSignalingNet.and.ListGDS =
which(!is.na(unlist(index.of.not.matching.preSignalingNet.and.ListGDS)))
```

```
index.of.not.matching.preSignalingNet.and.ListGDS

##  [1]   213  5245  5258  5271  5284  5297  5310  5323  5336  5349  5362
5375
## [13]  5388  8711  8715  8719 10057 10060 10077 10080 10081 10082 10083
10084
## [25] 10134 16235 17298 18206 18445 18447 18574 21348 21352 21356 25757

edgelist[index.of.not.matching.preSignalingNet.and.ListGDS[1:4],]

##           ID   Gene1   Gene2 Interaction type
## 213  E00213   HBEGF   HBEGF       activation
## 5245 E05245 ANAPC10 ANAPC10       activation
## 5258 E05258   CDC26   CDC26       activation
## 5271 E05271 ANAPC13 ANAPC13       activation
```

```
preSignalingNet1[[index.of.not.matching.preSignalingNet.and.ListGDS[2]]][,1:6
]
```

```
##           GSM1305903 GSM1305905 GSM1305907 GSM1305909 GSM1305902 GSM1305904
## ANAPC10        49.98      52.17      56.98      54.39      33.34      27.26
## ANAPC101       49.98      52.17      56.98      54.39      33.34      27.26
```

```
Eligibles =  (1:length(preSignalingNet2))[-
c(index.of.not.matching.preSignalingNet.and.ListGDS,idx.NO.gsms)]
length(Eligibles)
# There are 15805 edges for down stream analysis
```

Correlation analysis was performed on each edge (gene pair) after preprocessing step. Samples having value for the gene pairs, may come from different datasets and they should be separated and analyzed independently. Figure 6 represents the effect of this preprocessing on a gene pairs in the dataset. In panel A, because all the values are related to multiple datasets in preSignalingNet1[[4]], we can see a highly data dispersion. However, in the preSignalingNe2[[4]] all of these values are separated into 14 datasets. in panel B, six of these separated datasets are plotted.

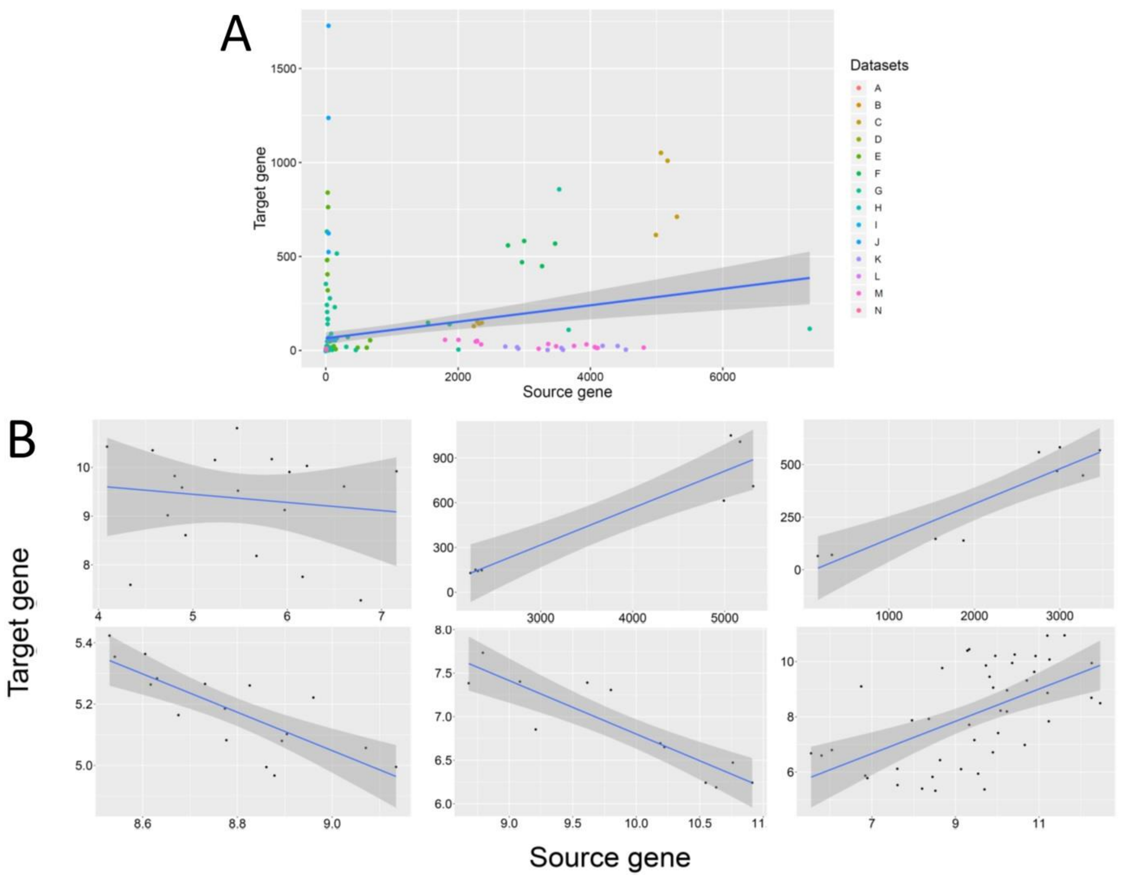


Figure 6

```
## null device
##           1
```

Getting rid of outliers

Values 1.5 times more than IQR are considered as outliers and are taken out.

```
for(i in Eligibles){

  if(class(preSignalingNet2[[i]])=="list"){
for(j in 1:length(preSignalingNet2[[i]]))
  outliers1 = boxplot.stats(as.numeric(preSignalingNet2[[i]][[j]][1,]))$out
  outliers2 = boxplot.stats(as.numeric(preSignalingNet2[[i]][[j]][2,]))$out

  index1 = which(as.numeric(preSignalingNet2[[i]][[j]][2,]) %in% outliers1)
  index2 = which(as.numeric(preSignalingNet2[[i]][[j]][2,]) %in% outliers2)
  index = unique(c(index1,index2))
  if(length(index) > 0){
  preSignalingNet2[[i]][[j]] = preSignalingNet2[[i]][[j]][,-index]
  }

  } else {
  outliers1 = boxplot.stats(as.numeric(preSignalingNet2[[i]][1,]))$out
  outliers2 = boxplot.stats(as.numeric(preSignalingNet2[[i]][2,]))$out

  index1 = which(as.numeric(preSignalingNet2[[i]][1,]) %in% outliers1)
  index2 = which(as.numeric(preSignalingNet2[[i]][2,]) %in% outliers2)
  index = unique(c(index1,index2))
  if(length(index) > 0){
    preSignalingNet2[[i]] = preSignalingNet2[[i]][,-index]}

  }
}
```

The following code proves that all the elements in preSignalingNet2 which is in the class dataframe, have more than two samples. The code must return a TRUE value.

```
CLASS = lapply(preSignalingNet2 , class)
all(lapply(lapply(preSignalingNet2[which(CLASS == "data.frame")] , dim),
function(x) x[2]) > 2)

## [1] FALSE
```

The following code presents the number of datasets which have two or fewer datasets.

```
sum(unlist(lapply(preSignalingNet2[which(CLASS == "list")] , function(x)
lapply(x,function(x) dim(x)[2] <= 2))))

## [1] 43
```

The following code shows the indices in preSignalingNet2 in the class list having datasets with less than three samples.

```
index = which(unlist(lapply(preSignalingNet2[which(CLASS == "list")] ,
function(x) any(unlist(lapply(x,function(x) dim(x)[2] < 3))))))
index

## E00598 E00868 E00877 E01073 E01081 E01430 E01439 E04464 E04490 E06633
E06640
##    393    550    556    683    689    914    921   2509   2527   3398
3405
## E06648 E06655 E06657 E06665 E06672 E06858 E06865 E06873 E06880 E06882
```

```
E06890
##   3413   3420   3422   3430   3436   3569   3576   3583   3589   3590
3596
## E06897 E07488 E07495 E07503 E07510 E07512 E07520 E07527 E09571 E09777
E09842
##   3602   4081   4087   4091   4096   4098   4103   4109   4999   5144
5193
## E13198 E16423 E16428 E17413 E17743 E18042 E18711 E25851 E25910 E25914
##   6901   7581   7586   8053   8216   8389   8787  11007  11050  11054
```

```
preSignalingNet2[which(CLASS == "list")][[index[1]]]
```

```
## [[1]]
##         GSM4312 GSM4317
## TGFB3   296.045 109.721
## TGFBR2   24.545 136.243
##
## [[2]]
##          GSM7621  GSM7622 GSM7623 GSM7624  GSM6681  GSM6682 GSM6683 GSM6684
## TGFB3    848.063  827.429 3403.71 2631.38  739.542  606.972 3023.23 2892.95
## TGFBR2 5545.760 5872.560 8368.38 7563.18 5431.500 5991.290 7298.31 6956.94
##          GSM6685  GSM6686 GSM6687 GSM6688
## TGFB3    715.008  824.526 3184.55 2850.80
## TGFBR2 4540.660 4318.670 7407.48 6698.04
```

Ineligible datasets are omitted from the preSignalingNet2 through the following code. Ineligible datasets are those which contain less than three samples.

```
for(i in 1:length(index)){
  idx = which(unlist(lapply(lapply(preSignalingNet2[which(CLASS ==
"list")][index][[i]] , dim) , function(x) x[2]<= 2)))
  preSignalingNet2[which(CLASS == "list")][index][[i]] =
preSignalingNet2[which(CLASS == "list")][index][[i]][-idx]
}
```

the following code is to check whether there is any element of preSignalingNet2 in class list with the length of one.

```
index = which(lapply(preSignalingNet2 , class) == "list" &
lapply(preSignalingNet2 , length) == 1)
index

## E00598
##    598

preSignalingNet2[[index]] = as.data.frame(preSignalingNet2[[index]])
```

Because some changes may be imposed on some elements, another checking is done to make sure all the datasets have more than two GSMs.

```
CLASS = lapply(preSignalingNet2 , class)
all(lapply(lapply(preSignalingNet2[which(CLASS == "data.frame")] , dim),
function(x) x[2]) > 2)

## [1] FALSE

all(unlist(lapply(preSignalingNet2[which(CLASS == "list")] , function(x)
any(unlist(lapply(x,function(x) dim(x)[2] >= 3))))))

## [1] TRUE
```

The elements which contain less than three GSMs are assigned empty.

```
index = which(lapply(lapply(preSignalingNet2[which(CLASS == "data.frame")] ,
dim), function(x) x[2]) < 3)
index

## E12586
##   2460

if(length(index)>0){
preSignalingNet2[which(CLASS == "data.frame")][[index]]
preSignalingNet2[which(CLASS == "data.frame")][[index]] = "empty"
}
```

Eligible edges are updated.

```
Eligibles =  which(lapply(preSignalingNet2 , class) == "list" |
lapply(preSignalingNet2 , class) == "data.frame")
length(Eligibles)

## [1] 15804
```

4.Correlation analysis for each edge

4.1.Creating a large list called SignalingNet which contains expression values and correlation information for each edge

The next step is to compute the correlation between the two interacting genes. Pearson, Spearman and Kendall correlation tests are calculated for each edge and a list called **SignalingNet** is created.

```
SignalingNet = preSignalingNet2

for(i in Eligibles){

  if(class(SignalingNet[[i]])=="list"){
  corAnalysis = list()
  dim = list()
  for(j in 1:length(SignalingNet[[i]])){

  p = cor.test(as.numeric(SignalingNet[[i]][[j]][1,]) ,
as.numeric(SignalingNet[[i]][[j]][2,]) , method = "pearson")
  s = cor.test(as.numeric(SignalingNet[[i]][[j]][1,]) ,
as.numeric(SignalingNet[[i]][[j]][2,]) , method = "spearman")
  k = cor.test(as.numeric(SignalingNet[[i]][[j]][1,]) ,
as.numeric(SignalingNet[[i]][[j]][2,]) , method = "kendall")
  pearson = as.data.frame(cbind(pearsoncor = unname(p$estimate) , pearsonpval
= p$p.value))
  spearman = as.data.frame(cbind(spearmancor = unname(s$estimate) ,
spearmanpval = s$p.value))
  kendall = as.data.frame(cbind(kendallcor = unname(k$estimate) , kendallpval
= k$p.value))
  corAnalysis[[j]] = list(pearson = pearson,spearman = spearman,kendall =
kendall)
  dim[[j]] = dim(SignalingNet[[i]][[j]])
  }

  SignalingNet[[i]] = list(Source = unname(edgelist[i,2]) , Target =
unname(edgelist[i,3])  , length = length(SignalingNet[[i]]) ,dim = dim ,
coExpr =  SignalingNet[[i]] ,
                             corAnalysis = corAnalysis , Interactiontype =
unname(edgelist[i,4]) , Coherency = "empty")

  } else {
    p = cor.test(as.numeric(SignalingNet[[i]][1,]) ,
as.numeric(SignalingNet[[i]][2,]) , method = "pearson")
    s = cor.test(as.numeric(SignalingNet[[i]][1,]) ,
as.numeric(SignalingNet[[i]][2,]) , method = "spearman")
    k = cor.test(as.numeric(SignalingNet[[i]][1,]) ,
as.numeric(SignalingNet[[i]][2,]) , method = "kendall")
    pearson = as.data.frame(cbind(pearsoncor = unname(p$estimate) ,
pearsonpval = p$p.value))
    spearman = as.data.frame(cbind(spearmancor = unname(s$estimate) ,
spearmanpval = s$p.value))
    kendall = as.data.frame(cbind(kendallcor = unname(k$estimate) ,
kendallpval = k$p.value))


    SignalingNet[[i]] = list(Source = unname(edgelist[i,2]) , Target =
unname(edgelist[i,3]) , length = 1 ,dim = dim(SignalingNet[[i]]) , coExpr =
SignalingNet[[i]] ,
                               corAnalysis = list(pearson = pearson,spearman
= spearman,kendall = kendall) ,
                               Interactiontype = unname(edgelist[i,4]) ,
Coherency = "empty")
  }

}



for(i in (1:length(SignalingNet))[-Eligibles]){
  SignalingNet[[i]] = list(Source = unname(edgelist[i,2]) , Target =
unname(edgelist[i,3]) , length = 0 ,dim = 0 , coExpr = NA , corAnalysis = NA
, Interactiontype = unname(edgelist[i,4]) , Coherency = "empty")
}
```

An example of eligible edge:

```
SignalingNet[[70]]

## $Source
## [1] "PDGFRB"
##
## $Target
## [1] "PLCG2"
##
## $length
## [1] 4
##
## $dim
## $dim[[1]]
## [1] 2 8
##
## $dim[[2]]
## [1] 2 8
##
## $dim[[3]]
## [1]  2 10
##
```

```
## $dim[[4]]
## [1]  2 30
##
##
## $coExpr
## $coExpr[[1]]
##        GSM490979 GSM490980 GSM490981 GSM490982 GSM490983 GSM490984
GSM490985
## PDGFRB   8.13064   8.35757   7.80769   8.03344   9.01956   8.54397
9.12372
## PLCG2    6.11117   6.29475   6.50859   6.23847   5.81704   5.82973
5.92684
##        GSM490986
## PDGFRB   9.15383
## PLCG2    6.08239
##
## $coExpr[[2]]
##        GSM244647 GSM244649 GSM244651 GSM244653 GSM244648 GSM244650
GSM244652
## PDGFRB   36.1238   39.6828   6.70644   13.8024   21.5159   34.3139
11.9610
## PLCG2   360.3680  373.0510  85.62970   68.4027  428.7340  424.1310
59.5258
##        GSM244654
## PDGFRB   12.7781
## PLCG2   104.2860
##
## $coExpr[[3]]
##        GSM102789 GSM102785 GSM102787 GSM102790 GSM102786 GSM102788
GSM102681
## PDGFRB   176.298   124.255   107.709   352.027   239.746   95.4943
1298.46
## PLCG2    565.279   259.747   554.512  1922.380   365.583 2091.8100
1613.89
##        GSM102783 GSM102782 GSM102784
## PDGFRB   780.864   951.725   846.224
## PLCG2   1324.060  1713.440  1369.790
##
## $coExpr[[4]]
##        GSM1143676 GSM1143677 GSM1143678 GSM1143679 GSM1143680 GSM1143681
## PDGFRB    7.31532    7.23638    7.54621    7.48674    7.67071    7.35978
## PLCG2     7.30048    7.22943    7.15666    6.98803    7.30929    6.96447
##        GSM1143682 GSM1143683 GSM1143684 GSM1143685 GSM1143686 GSM1143687
## PDGFRB    7.54044    7.39747    7.56700    7.40331    7.44638    7.82191
## PLCG2     7.11817    7.32129    7.08797    6.95786    7.06090    7.56741
##        GSM1143688 GSM1143689 GSM1143690 GSM1143691 GSM1143692 GSM1143693
## PDGFRB    7.54010    7.59168    7.78653    7.47773    7.25271    7.41038
## PLCG2     7.36197    7.49527    7.60299    7.37531    7.32986    7.18851
##        GSM1143694 GSM1143695 GSM1143696 GSM1143697 GSM1143698 GSM1143699
## PDGFRB    7.41699    7.27599    7.48022    7.45763    7.52638    7.78671
## PLCG2     7.19516    7.25258    7.05503    7.03926    7.35168    7.10651
##        GSM1143700 GSM1143701 GSM1143702 GSM1143703 GSM1143704 GSM1143705
## PDGFRB    7.52223    7.06359    7.76968    7.32825    7.41585    7.43055
## PLCG2     7.10267    7.24621    7.28328    7.25294    7.31378    7.46169
##
##
## $corAnalysis
## $corAnalysis[[1]]
## $corAnalysis[[1]]$pearson
##   pearsoncor pearsonpval adjusted.pearsonpval
## 1 -0.7402235  0.03573155            0.0790027
##
## $corAnalysis[[1]]$spearman
##   spearmancor spearmanpval
## 1  -0.7142857   0.05758929
##
## $corAnalysis[[1]]$kendall
##   kendallcor kendallpval
## 1       -0.5   0.1086806
##
##
## $corAnalysis[[2]]
## $corAnalysis[[2]]$pearson
##   pearsoncor pearsonpval adjusted.pearsonpval
## 1  0.8483993 0.007750156           0.02304932
##
## $corAnalysis[[2]]$spearman
##   spearmancor spearmanpval
## 1   0.6666667   0.08308532
##
## $corAnalysis[[2]]$kendall
##   kendallcor kendallpval
## 1  0.4285714    0.178869
##
##
## $corAnalysis[[3]]
## $corAnalysis[[3]]$pearson
##   pearsoncor pearsonpval adjusted.pearsonpval
## 1  0.4567883   0.1844641            0.2898644
##
## $corAnalysis[[3]]$spearman
##   spearmancor spearmanpval
## 1   0.2606061    0.4696753
##
## $corAnalysis[[3]]$kendall
##   kendallcor kendallpval
## 1  0.2444444   0.3807198
##
##
## $corAnalysis[[4]]
## $corAnalysis[[4]]$pearson
##   pearsoncor pearsonpval adjusted.pearsonpval
## 1  0.2661583   0.1551258            0.2530683
##
## $corAnalysis[[4]]$spearman
##   spearmancor spearmanpval
## 1   0.1764182    0.3495161
##
## $corAnalysis[[4]]$kendall
##   kendallcor kendallpval
## 1  0.1310345   0.3206742
##
##
##
## $Interactiontype
## [1] "activation"
##
## $Coherency
## [1] "empty"
```

An example of ineligible edge:

```
SignalingNet[[43]]
```

```
## $Source
## [1] "IGF1R"
##
## $Target
## [1] "PLCG1"
##
## $length
## [1] 0
##
## $dim
## [1] 0
##
## $coExpr
## [1] NA
##
## $corAnalysis
## [1] NA
##
## $Interactiontype
## [1] "activation"
##
## $Coherency
## [1] "empty"
```

Using the following code, all the Pearson, Spearman and Kendall correlation coefficients and p-values are saved in the following objects.

```r
pearsonpvalues = c()
spearmanpvalues = c()
kendallpvalues = c()
pearsoncors = c()
spearmancors = c()
kendallcors = c()


for(i in Eligibles){
  if(SignalingNet[[i]]$length > 1){
    for(j in 1:length(SignalingNet[[i]]$corAnalysis)){

        a = SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsonpval
        pearsonpvalues = c(pearsonpvalues,a)
        b = SignalingNet[[i]]$corAnalysis[[j]]$spearman$spearmanpval
        spearmanpvalues = c(spearmanpvalues,b)
        c = SignalingNet[[i]]$corAnalysis[[j]]$kendall$kendallpval
        kendallpvalues = c(kendallpvalues,c)
        d = SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor
        pearsoncors = c(pearsoncors,d)
        e = SignalingNet[[i]]$corAnalysis[[j]]$spearman$spearmancor
        spearmancors = c(spearmancors,e)
        f = SignalingNet[[i]]$corAnalysis[[j]]$kendall$kendallcor
        kendallcors = c(kendallcors,f)


    }
  }else {

      a = SignalingNet[[i]]$corAnalysis$pearson$pearsonpval
      pearsonpvalues = c(pearsonpvalues,a)
      b = SignalingNet[[i]]$corAnalysis$spearman$spearmanpval
      spearmanpvalues = c(spearmanpvalues,b)
      c = SignalingNet[[i]]$corAnalysis$kendall$kendallpval
      kendallpvalues = c(kendallpvalues,c)
      d = SignalingNet[[i]]$corAnalysis$pearson$pearsoncor
      pearsoncors = c(pearsoncors,d)
      e = SignalingNet[[i]]$corAnalysis$spearman$spearmancor
      spearmancors = c(spearmancors,e)
      f = SignalingNet[[i]]$corAnalysis$kendall$kendallcor
      kendallcors = c(kendallcors,f)
    }

}

length(pearsonpvalues)
```

```
## [1] 76897

length(spearmanpvalues)

## [1] 76897

length(kendallpvalues)

## [1] 76897
```

Using the codes below, p-values are adjusted using “fdr” method.

```
adjusted.pearsonpvalues = p.adjust(pearsonpvalues , method = "fdr")

all(lapply(SignalingNet[Eligibles] , function(x) x$length) > 0)

## [1] TRUE
```

Adjusted p-values are added to the **SignalingNet** object. After completing this operation, adjusted.pearsonpvalues object must be NULL.

```
for(i in Eligibles){
  l = SignalingNet[[i]]$length
  if(l > 1){
for(j in 1:l){
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval =
adjusted.pearsonpvalues[j]
}
 adjusted.pearsonpvalues = adjusted.pearsonpvalues[-1:-l]
 }else{
    SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval =
adjusted.pearsonpvalues[1]
    adjusted.pearsonpvalues = adjusted.pearsonpvalues[-1]
  }
}
```

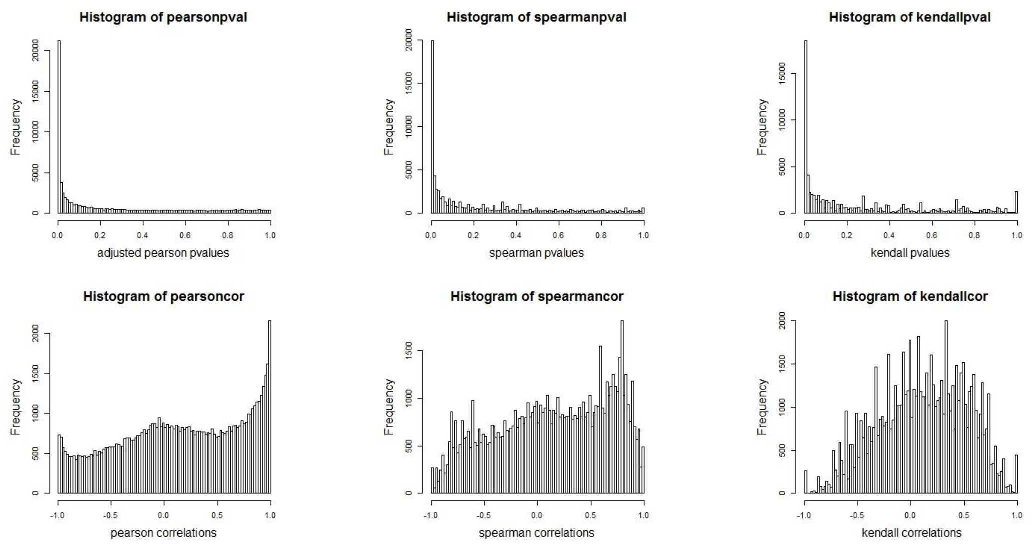


Figure7

In the histograms, most of the p-values are under 0.1. The number of positive coefficients are larger than negative coefficients. so, histograms of coefficients are left-skewed.

The code show that how many edges have a p-value less than a specified value.

```
cdfpearsonpval = c()
length(pearsonpvalues)

## [1] 76897

pvalue = seq(0.01 , 0.99 , by = 0.01)
for(i in 1:99){
  cdfpearsonpval[i] = sum(na.omit(pearsonpvalues < pvalue[i]))
}
cdfpearsonpval = cdfpearsonpval/length(pearsonpvalues)
cdfpearsonpval = rbind(pvalue,cdfpearsonpval)
cdfpearsonpval[,1:5]

##                     [,1]      [,2]      [,3]      [,4]      [,5]
## pvalue         0.0100000 0.0200000 0.0300000 0.0400000 0.0500000
## cdfpearsonpval 0.3521854 0.4038779 0.4370782 0.4620987 0.4842452

# For example in the codes below, 35 percent of edges had a pvalue less than
0.01
```

```
cdfspearmanpval = c()
pvalue = seq(0.01 , 0.99 , by = 0.01)
for(i in 1:99){
  cdfspearmanpval[i] = sum(na.omit(spearmanpvalues < pvalue[i]))
}
cdfspearmanpval = cdfspearmanpval/length(spearmanpvalues)
cdfspearmanpval = rbind(pvalue,cdfspearmanpval)
cdfspearmanpval[,1:5]

##                     [,1]      [,2]      [,3]      [,4]     [,5]
## pvalue          0.010000 0.0200000 0.0300000 0.0400000 0.050000
## cdfspearmanpval 0.259477 0.3160461 0.3524975 0.3870892 0.410094

cdfkendallpval = c()
pvalue = seq(0.01 , 0.99 , by = 0.01)
for(i in 1:99){
  cdfkendallpval[i] = sum(na.omit(kendallpvalues < pvalue[i]))
}
cdfkendallpval = cdfkendallpval/length(kendallpvalues)
cdfkendallpval = rbind(pvalue,cdfkendallpval)
cdfkendallpval[,1:5]

##                     [,1]      [,2]      [,3]      [,4]      [,5]
## pvalue         0.0100000 0.0200000 0.0300000 0.0400000 0.0500000
## cdfkendallpval 0.2406726 0.2939256 0.3237578 0.3505208 0.3763086
```

```
cdfpearsoncorr = c()
length(pearsoncors)

## [1] 76897

positiveCorr = seq(0 , 0.9 , by = 0.1)
for(i in 1:10){
  cdfpearsoncorr[i] = sum(na.omit(pearsoncors > positiveCorr[i]))
}
cdfpearsoncorr = cdfpearsoncorr/length(pearsoncors)
cdfpearsoncorr = rbind(positiveCorr,cdfpearsoncorr)
# about 10% of edges have pearson corr larger than 0.9
cdfpearsoncorr

##                     [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## positiveCorr   0.0000000 0.1000000 0.2000000 0.3000000 0.4000000 0.5000000
## cdfpearsoncorr 0.5919607 0.5378103 0.4845313 0.4329558 0.3831749 0.3341223
##                     [,7]      [,8]      [,9]     [,10]
## positiveCorr   0.6000000 0.7000000 0.8000000 0.9000000
## cdfpearsoncorr 0.2848486 0.2310753 0.1724905 0.1018245
```

```
cdfpearsoncorr1 = c()
length(pearsoncors)

## [1] 76897
```

```
negativeCorr = seq(0 , -0.9 , by = -0.1)
for(i in 1:10){
  cdfpearsoncorr1[i] = sum(na.omit(pearsoncors < negativeCorr[i]))
}
cdfpearsoncorr1 = cdfpearsoncorr1/length(pearsoncors)
cdfpearsoncorr1 = rbind(negativeCorr,cdfpearsoncorr1)
# about 3.9% of edges have pearson corr less than -0.9
cdfpearsoncorr1
```

```
##                        [,1]       [,2]       [,3]       [,4]       [,5]
[,6]
## negativeCorr    0.0000000 -0.1000000 -0.2000000 -0.3000000 -0.4000000 -
0.500000
## cdfpearsoncorr1 0.4064528  0.3499096  0.2969817  0.2505169  0.2071992
0.168212
##                         [,7]        [,8]        [,9]       [,10]
## negativeCorr    -0.6000000 -0.70000000 -0.80000000 -0.90000000
## cdfpearsoncorr1  0.1320208  0.09957476  0.06936551  0.03939035
```

```
cdfspearmancorr = c()
positiveCorr = seq(0 , 0.9 , by = 0.1)
for(i in 1:10){
  cdfspearmancorr[i] = sum(na.omit(spearmancors > positiveCorr[i]))
}
cdfspearmancorr = cdfspearmancorr/length(spearmancors)
cdfspearmancorr = rbind(positiveCorr,cdfspearmancorr)
# about 3.5% of edges have spearman corr larger than 0.9
cdfspearmancorr
```

```
##                        [,1]      [,2]      [,3]      [,4]      [,5]
[,6]
## positiveCorr    0.0000000 0.1000000 0.2000000 0.3000000 0.4000000
0.5000000
## cdfspearmancorr 0.6002185 0.5422188 0.4857667 0.4329688 0.3782202
0.3209748
##                        [,7]      [,8]      [,9]      [,10]
## positiveCorr    0.6000000 0.7000000 0.8000000 0.90000000
## cdfspearmancorr 0.2557057 0.1897083 0.1024227 0.03524195
```

```
cdfspearmancorr1 = c()
negativeCorr = seq(0 , -0.9 , by = -0.1)
for(i in 1:10){
  cdfspearmancorr1[i] = sum(na.omit(spearmancors < negativeCorr[i]))
}
cdfspearmancorr1 = cdfspearmancorr1/length(spearmancors)
cdfspearmancorr1 = rbind(negativeCorr,cdfspearmancorr1)
```

```
# about 1.2% of edges have spearman core less than -0.9
cdfspearmancorr1

##                        [,1]      [,2]       [,3]       [,4]       [,5]
[,6]
## negativeCorr     0.00000 -0.100000 -0.2000000 -0.3000000 -0.4000000 -
0.5000000
## cdfspearmancorr1 0.39523  0.339155  0.2878396  0.2420901  0.2003329
0.1601883
##                           [,7]        [,8]        [,9]       [,10]
## negativeCorr     -0.6000000 -0.70000000 -0.80000000 -0.90000000
## cdfspearmancorr1  0.1183401  0.08026321  0.03743969  0.01204208
```

```
cdfkendallcorr = c()
positiveCorr = seq(0 , 0.9 , by = 0.1)
for(i in 1:10){
  cdfkendallcorr[i] = sum(na.omit(kendallcors > positiveCorr[i]))
}
cdfkendallcorr = cdfkendallcorr/length(kendallcors)
cdfkendallcorr = rbind(positiveCorr,cdfkendallcorr)
# about 0.8% of edges have kendall corr larger than 0.9
cdfkendallcorr

##                      [,1]      [,2]      [,3]      [,4]     [,5]      [,6]
## positiveCorr   0.000000 0.1000000 0.2000000 0.3000000 0.400000 0.5000000
## cdfkendallcorr 0.595966 0.5151306 0.4342172 0.3588176 0.277228 0.1983823
##                       [,7]       [,8]       [,9]       [,10]
## positiveCorr   0.6000000 0.70000000 0.80000000 0.900000000
## cdfkendallcorr 0.1231387 0.06464491 0.02395412 0.008517888
```

```
cdfkendallcorr1 = c()
negativeCorr = seq(0 , -0.9 , by = -0.1)
for(i in 1:10){
  cdfkendallcorr1[i] = sum(na.omit(kendallcors < negativeCorr[i]))
}
cdfkendallcorr1 = cdfkendallcorr1/length(kendallcors)
cdfkendallcorr1 = rbind(negativeCorr,cdfkendallcorr1)
# about 0.32% of edges have kendall core less than -0.9
cdfkendallcorr1

##                       [,1]       [,2]       [,3]       [,4]       [,5]
## negativeCorr    0.0000000 -0.1000000 -0.2000000 -0.3000000 -0.4000000
## cdfkendallcorr1 0.3903793  0.3138874  0.2426232  0.1856639  0.1293549
##                           [,6]        [,7]        [,8]        [,9]
[,10]
## negativeCorr    -0.50000000 -0.60000000 -0.70000000 -0.80000000 -
0.900000000
```

```
## cdfkendallcorr1  0.08337126  0.04706295  0.02379807  0.00923313
0.004200424
```

Here we create a function for our database “SignalingNet” to check the number of its elements having less than n GSMs amonge eligible elements.

```
GSMnumber = function(x){logic = c()

for(i in Eligibles){
  if(SignalingNet[[i]]$length > 1){
  logic[i] = dim(do.call("cbind" , SignalingNet[[i]]$coExpr))[2] < x
  }else{logic[i] = dim(SignalingNet[[i]]$coExpr)[2] < x}
}
length(which(logic))
}

# to show the number of edges that have less than 10 GSMs:
GSMnumber(10)

## [1] 1567
```

5.One edge Subgraphs

5.1.Getting the number of Activation and Inhibition edges having p-values and correlations of interest.

In this section the number of edges having the specific p-value and correlation are computed. The first condition determines if SignalingNet[[i]]$coranalysis has one element or multiple elements. If there is more than one element, all the elements should be non-NA. Then we check all the elements in SignalingNet[[i]] to have p-value < 0.05 and correlation > 0 . If all the conditions are met, the index of the edge is stored at **index.pval.cor1** object.

```
index.pval.cor1 = c()

for(i in Eligibles){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    logic1=c()
    for(j in 1:SignalingNet[[i]]$length){
      logic1[j] =
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic1)){

  for(j in 1:SignalingNet[[i]]$length){

  logic[j] = SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval
< 0.05 & SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
  }
  if(all(logic)) {
    index.pval.cor1 = c(index.pval.cor1,i)
  }
  }
  } else { if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
    a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
    if(a){
      index.pval.cor1 = c(index.pval.cor1,i)
      }
  }
  }

}
```

The first condition determines if there are more than one GDS for each edge. The second condition tells we need edges for which all the Pearson p-values are NA or > 0.05. If all the conditions are met, the index of the edge is stored at **index.pval.cor.NA** object.

```
index.pval.cor.NA = c()

for(i in Eligibles){
  logic = c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){

      logic[j]=any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
| SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
    }
    if(all(logic)){ index.pval.cor.NA = c(index.pval.cor.NA,i) }

  }else{ if(any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))|
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05){
    index.pval.cor.NA = c(index.pval.cor.NA,i)}
  }
}
```

The first condition determines whether SignalingNet[[i]]$coranalysis has one element or multiple elements. If there are several elements, all of them should be non-NA. Then we check that all elements in SignalingNet[[i]] to have the p-value < 0.05 or correlation < 0 . If all the conditions are met, the index of the edge will be stored at **index.pval.cor2** object.

```
index.pval.cor2 = c()

for(i in Eligibles){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    logic1=c()
    for(j in 1:SignalingNet[[i]]$length){
      logic1[j] =
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic1)){

      for(j in 1:SignalingNet[[i]]$length){

        logic[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic)) {
        index.pval.cor2 = c(index.pval.cor2,i)
      }
    }
  } else { if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
    a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
    if(a){
      index.pval.cor2 = c(index.pval.cor2,i)
    }
  }
  }

}
```

5.2.Determining coherency of edges

If an edge is activation, and p-value < 0.05 and Pearson correlation > 0, or If an edge is inhibition, and p-value < 0.05 and pearson correlation < 0, the edge is coherent. If edge is activation, and p-value < 0.05 and pearson correlation < 0, or If edge is inhibition, and p-value < 0.05 and Pearson correlation > 0, the edge is incoherent. If p-value > 0.05 or p-value == NA, the edge is NA.

```
logic = edgelist[index.pval.cor1,4] == "activation"
length(which(logic))

## [1] 896

idx.Act1 = which(logic)


logic = edgelist[index.pval.cor1,4] == "inhibition"
length(which(logic))

## [1] 243

idx.Inh1 = which(logic)
```

```
for(i in index.pval.cor1){
  if(SignalingNet[[i]][[7]] == "activation"){
    SignalingNet[[i]][[8]] = "Coherent"
  } else {SignalingNet[[i]][[8]] = "Incoherent"}
}
```

```
logic = edgelist[index.pval.cor2,4] == "activation"
length(which(logic))

## [1] 457

idx.Act2 = which(logic)


logic = edgelist[index.pval.cor2,4] == "inhibition"
length(which(logic))

## [1] 130

idx.Inh2 = which(logic)
```

```
for(i in index.pval.cor2){
  if(SignalingNet[[i]][[7]] == "activation"){
    SignalingNet[[i]][[8]] = "Incoherent"
  } else {SignalingNet[[i]][[8]] = "Coherent"}
}
```

```
logic = edgelist[index.pval.cor.NA,4] == "activation"
length(which(logic))

## [1] 4602

idx.Act.NA = which(logic)

logic = edgelist[index.pval.cor.NA,4] == "inhibition"
length(which(logic))

## [1] 1246

idx.Inh.NA = which(logic)
```

```
for(i in index.pval.cor.NA){
  SignalingNet[[i]][[8]] = NA
}
```

```
number.of.coherent.edges=0
number.of.incoherent.edges=0
number.of.NA.edges=0

for(i in c(index.pval.cor1,index.pval.cor2)){
  if(SignalingNet[[i]][[8]] == "Coherent"){number.of.coherent.edges =
number.of.coherent.edges + 1
  }else if(SignalingNet[[i]][[8]] == "Incoherent"){number.of.incoherent.edges
= number.of.incoherent.edges + 1
  }
}


for(i in index.pval.cor.NA){
   if(is.na(SignalingNet[[i]][[8]])){number.of.NA.edges = number.of.NA.edges
+ 1}
}

number.of.coherent.edges

## [1] 1026

number.of.incoherent.edges

## [1] 700

number.of.NA.edges

## [1] 5848
```

In the following code, some eligible edges which are heterogeneous are stored at the object called **index.heterogeneous**. Heterogeneous edges are those for which there are multiple datasets with dissimilar p-values and correlations (Table 2).

```
index.heterogeneous = Eligibles[!(Eligibles %in%  index)]

length(index.heterogeneous )

## [1] 15803

length(c(index.pval.cor1 , index.pval.cor2 , index.pval.cor.NA ,
index.heterogeneous ))

## [1] 23377

length(unique(c(index.pval.cor1 , index.pval.cor2 , index.pval.cor.NA ,
index.heterogeneous )))

## [1] 15805

length(Eligibles)

## [1] 15804

logic = edgelist[index.heterogeneous,4] == "activation"
length(which(logic))

## [1] 12693

logic = edgelist[index.heterogeneous,4] == "inhibition"
length(which(logic))

## [1] 3110
```

6. Unconnected gene pairs

For all the four independent analyses, we randomly select 1000 unconnected gene pairs through adjacency matrix self-multiplication. After that, correlation analysis is done on these gene pairs. To do that, a graph is created from the eligible edges in the edgelist, and a non-weighted adjacency matrix is created from the giant component of the graph. For more information refer to the section 8. If an entry between two genes in an adjacency matrix stays zero in all the multiplications, there is no path which could connect these two genes.

```
matpower2 = AdjMatrix  %*% AdjMatrix

matpower3 = matpower2  %*% AdjMatrix

matpower4 = matpower3  %*% AdjMatrix

matpower5 = matpower4  %*% AdjMatrix

matpower6 = matpower5 %*% AdjMatrix

matpower7 = matpower6 %*% AdjMatrix

matpower8 = matpower7 %*% AdjMatrix

matpower9 = matpower8  %*% AdjMatrix

matpower10 = matpower9  %*% AdjMatrix

matpower11 = matpower10  %*% AdjMatrix

matpower12 = matpower11  %*% AdjMatrix

matpower13 = matpower12 %*% AdjMatrix

matpower14 = matpower13 %*% AdjMatrix

matpower15 = matpower14  %*% AdjMatrix

matpower16 = matpower15  %*% AdjMatrix

matpower17 = matpower16  %*% AdjMatrix

matpower18 = matpower17  %*% AdjMatrix

matpower19 = matpower18  %*% AdjMatrix

matpower20 = matpower19  %*% AdjMatrix
```

```
matpower21 = matpower20  %*% AdjMatrix

matpower22 = matpower21  %*% AdjMatrix

matpower23 = matpower22  %*% AdjMatrix

matpower24 = matpower23  %*% AdjMatrix

matpower25 = matpower24  %*% AdjMatrix

matpower26 = matpower25  %*% AdjMatrix

matpower27 = matpower26  %*% AdjMatrix

matpower28 = matpower27  %*% AdjMatrix

matpower29 = matpower28  %*% AdjMatrix

matpower30 = matpower29  %*% AdjMatrix

matpower31 = matpower30  %*% AdjMatrix

matpower32 = matpower31  %*% AdjMatrix

matpower33 = matpower32  %*% AdjMatrix

matpower34 = matpower33  %*% AdjMatrix

matpower35 = matpower34  %*% AdjMatrix

matpower36 = matpower35  %*% AdjMatrix

matpower37 = matpower36  %*% AdjMatrix

matpower38 = matpower37  %*% AdjMatrix

matpower39 = matpower38  %*% AdjMatrix

matpower40 = matpower39  %*% AdjMatrix
```

```
rn = rownames(AdjMatrix)
cn = colnames(AdjMatrix)
lzeros = list()
```

In the following code, gene pairs for which the value is zero in all 40 non-weighted adjacency matrices are selected.

```r
for(i in 1:length(matpower40[1,])){
  a = cbind(1,2)
  for(j in 1:length(matpower40[,1])){
    if(matpower2[i,j] == 0 & matpower3[i,j] == 0 & matpower4[i,j] == 0 &
matpower5[i,j] == 0 &
       matpower6[i,j] == 0 & matpower7[i,j] == 0 & matpower8[i,j] == 0 &
matpower9[i,j] == 0 &
       matpower10[i,j] == 0 & matpower11[i,j] == 0 & matpower12[i,j] == 0 &
matpower13[i,j] == 0 &
       matpower14[i,j] == 0 & matpower15[i,j] == 0 & matpower16[i,j] == 0 &
matpower17[i,j] == 0 &
       matpower18[i,j] == 0 & matpower19[i,j] == 0 & matpower20[i,j] == 0 &
matpower21[i,j] == 0 &
       matpower22[i,j] == 0 & matpower23[i,j] == 0 & matpower24[i,j] == 0 &
matpower25[i,j] == 0 &
       matpower26[i,j] == 0 & matpower27[i,j] == 0 & matpower28[i,j] == 0 &
matpower29[i,j] == 0 &
       matpower30[i,j] == 0 & matpower31[i,j] == 0 & matpower32[i,j] == 0 &
matpower33[i,j] == 0 &
       matpower34[i,j] == 0 & matpower35[i,j] == 0 & matpower36[i,j] == 0 &
matpower37[i,j] == 0 &
       matpower38[i,j] == 0 & matpower39[i,j] == 0 & matpower40[i,j] == 0 &
AdjMatrix[i,j] == 0 ){

      a = rbind(cbind(rn[i],cn[j]),a)
    }
  }

  lzeros[[i]] = a
  print(i)
}


unconnected.edgelist = do.call("rbind" , lzeros)
index = which(unconnected.edgelist[,1]=="1")
unconnected.edgelist=unconnected.edgelist[-index,]
```

There should be no common edges between the main edgelist and unconnected-gene-pair edgelist. because matpower40 adjacency matrix was driven from the giant component of the graph which was created from eligible-edge edgelist, there may exist some common edges between unconnected.edgelist and the eligible-edge edgelist. The reason why this happens is that some eligible edges are not included in the giant component. These edges should be omitted from the unconnected.edgelist. After that, we randomly selected 2000 edges from this edgelist.

```
e = edgelist[Eligibles,2:3]


index.of.rows.in.x.that.are.in.y  <- function(x,y)
{
  x.vec <- apply(x, 1, paste, collapse = "")
  y.vec <- apply(y, 1, paste, collapse = "")
  index = x.vec %in% y.vec
  return(which(index))
}
Index = index.of.rows.in.x.that.are.in.y(unconnected.edgelist,e)
Index

# unconnected.edgelist = unconnected.edgelist[-Index,]

short.unconnected.edgelist = unconnected.edgelist[seq(1,4000000,2000),]
```

The loop-form gene pairs are removed through the following code.

```
#
logic=c()
for(i in 1:2000){
  logic[i] = short.unconnected.edgelist[i,1]==short.unconnected.edgelist[i,2]
}
idxloop = which(logic)
idxloop
short.unconnected.edgelist[idxloop,]
short.unconnected.edgelist = short.unconnected.edgelist[-idxloop,]
```

All the gene names of each dataset are stored as an element of **list.gene** object.

```
list.genes = lapply(ListGDS , function(x) unname(x[,1]))
```

Here, a large list called **pre.unconnected.SignalingNet1** is created. Each element of this list is representative of each gene pair in **short.unconnected.edgelist**, and contains gene expression profiles for that gene pair. Some gene pairs are in multiple datasets and some of them are in one dataset. So, some elements in pre.unconnected.SignalingNet1 contain a list of multiple datasets and some of them contain just one dataset. To construct this object the following instruction is applied using **ListGDS** object:

The first condition is to see which datasets contain gene pairs in edge i in the short.unconnected.edgelist. The second condition tells if there are more than one datasets which satisfy the previous condition. This condition separates each element of pre.unconnected.SignalingNet1 into multiple datasets. The third condition tells that in dataset j, just two rows are related to the gene pairs. If the third condition is not satisfied, then between the similar-gene-named rows, we select the one with the largest IQR.

```
pre.unconnected.SignalingNet1 = list()
```

```
for(i in 1:length(short.unconnected.edgelist[,1])){
  index = which(unlist(lapply(list.genes , function(x)
all(short.unconnected.edgelist[i,] %in% x))))
  if(length(index) > 0){

    small.list = list()
    for(j in 1:length(index)){
    d = ListGDS[[index[j]]][which(list.genes[[index[j]]] %in%
short.unconnected.edgelist[i,]) , ]

    if(length(d[,1])==2){
      rownames(d) = d[,1]
      d = d[,-1]
      small.list[[j]] = d

    }else{
      n = d[,1]
      n1 = unique(n)

      index1 = which(n %in% n1[1])
      index2 = which(n %in% n1[2])

      if(length(index1)>1){
        d1 = d[index1,]

        IQRs = c()
        for(z in 1:length(d1[,1])){
          a=as.numeric(na.omit(as.numeric(d1[z,-1])))
          IQRs[z] = quantile(a, 0.75 ) - quantile(a, 0.25)
        }
        row.one = d1[which.max(IQRs),]

      }else{
        row.one = d[which(d[,1] == n1[1]),]
      }


      if(length(index2)>1){
        d2 = d[index2,]

        IQRs = c()
        for(z in 1:length(d2[,1])){
          a=as.numeric(na.omit(as.numeric(d2[z,-1])))
          IQRs[z] = quantile(a, 0.75 ) - quantile(a, 0.25)
        }
        row.two = d2[which.max(IQRs),]

      }else{
        row.two = d[which(d[,1] == n1[2]),]
      }

      dNEW = rbind(row.one,row.two)
      rownames(dNEW) = dNEW[,1]
      dNEW = dNEW[,-1]
      small.list[[j]] = dNEW

    }
    }

    if(j > 1){
    pre.unconnected.SignalingNet1[[i]] = small.list
    }else{ pre.unconnected.SignalingNet1[[i]] = small.list[[1]]}

  }
  #print(i)
}
```

Some elements in pre.unconnected.SignalingNet1 are a list of multiple datasets and are in class list, some elements are a dataset and are in class dataframe, and some elements of are NULL. All the elements of this object containing datasets are stored in **pre.unconnected.SignalingNet2** object.

```
CLASS = lapply(pre.unconnected.SignalingNet1 , class)
index = which(CLASS == "list" |  CLASS == "data.frame")


pre.unconnected.SignalingNet2 = pre.unconnected.SignalingNet1[index]
```

Getting rid of outliers

```
for(i in 1:length(pre.unconnected.SignalingNet2)){
  if(class(pre.unconnected.SignalingNet2[[i]])=="list"){
    for(j in 1:length(pre.unconnected.SignalingNet2[[i]]))
      outliers1 =
boxplot.stats(as.numeric(pre.unconnected.SignalingNet2[[i]][[j]][1,]))$out
    outliers2 =
boxplot.stats(as.numeric(pre.unconnected.SignalingNet2[[i]][[j]][2,]))$out

    index1 = which(as.numeric(pre.unconnected.SignalingNet2[[i]][[j]][2,])
%in% outliers1)
    index2 = which(as.numeric(pre.unconnected.SignalingNet2[[i]][[j]][2,])
%in% outliers2)
    index = unique(c(index1,index2))
    if(length(index) > 0){
      pre.unconnected.SignalingNet2[[i]][[j]] =
pre.unconnected.SignalingNet2[[i]][[j]][,-index]
    }

  } else {
    outliers1 =
boxplot.stats(as.numeric(pre.unconnected.SignalingNet2[[i]][1,]))$out
    outliers2 =
boxplot.stats(as.numeric(pre.unconnected.SignalingNet2[[i]][2,]))$out

    index1 = which(as.numeric(pre.unconnected.SignalingNet2[[i]][1,]) %in%
outliers1)
    index2 = which(as.numeric(pre.unconnected.SignalingNet2[[i]][2,]) %in%
outliers2)
    index = unique(c(index1,index2))
    if(length(index) > 0){
      pre.unconnected.SignalingNet2[[i]] =
pre.unconnected.SignalingNet2[[i]][,-index]}

  }
}
```

```
CLASS = lapply(pre.unconnected.SignalingNet2 , class)


# The following code proves that all the elements in
pre.unconnected.SignalingNet2 which is data frame, having more than two
samples.
all(lapply(lapply(pre.unconnected.SignalingNet2[which(CLASS == "data.frame")]
, dim), function(x) x[2]) > 2)
```

```r
lapply(pre.unconnected.SignalingNet2[which(CLASS == "list")] , function(x)
lapply(x,dim))


# The following code present the number of datasets with less than three
samples.
sum(unlist(lapply(pre.unconnected.SignalingNet2[which(CLASS == "list")] ,
function(x) lapply(x,function(x) dim(x)[2] <= 2))))


# The following code shows the indices of pre.unconnected.SignalingNet2 which
are in the class list having datasets with less than three samples.
index = which(!(unlist(lapply(pre.unconnected.SignalingNet2[which(CLASS ==
"list")] , function(x) all(unlist(lapply(x,function(x) dim(x)[2] > 2)))))))
index
# 92 216 491 582


# We deleted the ineligible datasets from the pre.unconnected.SignalingNet2
through the following code
for(i in 1:length(index)){
  idx = which(unlist(lapply(lapply(pre.unconnected.SignalingNet2[which(CLASS
== "list")][index][[i]] , dim) , function(x) x[2]<= 2)))
  pre.unconnected.SignalingNet2[which(CLASS == "list")][index][[i]] =
pre.unconnected.SignalingNet2[which(CLASS == "list")][index][[i]][-idx]
  }



# Through the following code we check whether or not there is any element in
pre.unconnected.SignalingNet2 in the class list having the length of one
(have just one dataset)
any(lapply(pre.unconnected.SignalingNet2[which(CLASS == "list")] , length) ==
1)
```

Correlation analysis

A new list called **unconnected.SignalingNet** is created by the following code involving the correlation results.

```r
unconnected.SignalingNet = pre.unconnected.SignalingNet2


for(i in (1:length(unconnected.SignalingNet))){
  if(class(unconnected.SignalingNet[[i]])=="list"){
    corAnalysis = list()
    dim = list()
    for(j in 1:length(unconnected.SignalingNet[[i]])){

      p = cor.test(as.numeric(unconnected.SignalingNet[[i]][[j]][1,]) ,
as.numeric(unconnected.SignalingNet[[i]][[j]][2,]) , method = "pearson")
      s = cor.test(as.numeric(unconnected.SignalingNet[[i]][[j]][1,]) ,
as.numeric(unconnected.SignalingNet[[i]][[j]][2,]) , method = "spearman")
      k = cor.test(as.numeric(unconnected.SignalingNet[[i]][[j]][1,]) ,
as.numeric(unconnected.SignalingNet[[i]][[j]][2,]) , method = "kendall")
      pearson = as.data.frame(cbind(pearsoncor = unname(p$estimate) ,
pearsonpval = p$p.value))
      spearman = as.data.frame(cbind(spearmancor = unname(s$estimate) ,
spearmanpval = s$p.value))
      kendall = as.data.frame(cbind(kendallcor = unname(k$estimate) ,
kendallpval = k$p.value))
      corAnalysis[[j]] = list(pearson = pearson,spearman = spearman,kendall =
kendall)
      dim[[j]] = dim(unconnected.SignalingNet[[i]][[j]])
    }

    unconnected.SignalingNet[[i]] = list(Source =
rownames(unconnected.SignalingNet[[i]][[1]])[1] , Target =
rownames(unconnected.SignalingNet[[i]][[1]])[2]  , length =
length(unconnected.SignalingNet[[i]]) ,dim = dim , coExpr =
unconnected.SignalingNet[[i]] ,
                             corAnalysis = corAnalysis , Interactiontype =
"None" , Coherency = "empty")

  } else {
    p = cor.test(as.numeric(unconnected.SignalingNet[[i]][1,]) ,
as.numeric(unconnected.SignalingNet[[i]][2,]) , method = "pearson")
    s = cor.test(as.numeric(unconnected.SignalingNet[[i]][1,]) ,
as.numeric(unconnected.SignalingNet[[i]][2,]) , method = "spearman")
    k = cor.test(as.numeric(unconnected.SignalingNet[[i]][1,]) ,
as.numeric(unconnected.SignalingNet[[i]][2,]) , method = "kendall")
    pearson = as.data.frame(cbind(pearsoncor = unname(p$estimate) ,
pearsonpval = p$p.value))
    spearman = as.data.frame(cbind(spearmancor = unname(s$estimate) ,
spearmanpval = s$p.value))
    kendall = as.data.frame(cbind(kendallcor = unname(k$estimate) ,
kendallpval = k$p.value))


    unconnected.SignalingNet[[i]] = list(Source =
unconnected.SignalingNet[[i]])[1] , Target =
unconnected.SignalingNet[[i]])[2] , length = 1 ,dim =
dim(unconnected.SignalingNet[[i]]) , coExpr =  unconnected.SignalingNet[[i]]
,
                             corAnalysis = list(pearson = pearson,spearman =
spearman,kendall = kendall) ,
                             Interactiontype = "None" , Coherency = "empty")
  }
  #print(i)
}
```

```
# The first 1000 elements of unconnected.SignalingNet object are selected.
unconnected.SignalingNet = unconnected.SignalingNet[1:1000]

unconnected.SignalingNet[[1]]

## $Source
## [1] "EGFR"
##
## $Target
## [1] "GDF7"
##
## $length
## [1] 3
##
## $dim
## $dim[[1]]
## [1] 2 6
##
## $dim[[2]]
## [1]  2 10
##
## $dim[[3]]
## [1]  2 12
##
##
## $coExpr
## $coExpr[[1]]
##      GSM148690 GSM148691 GSM148692 GSM148687 GSM148688 GSM148689
## EGFR   7.43996   8.09821   7.97677   6.60359   6.57079    6.6735
## GDF7   9.69006   9.43124   9.50077   7.13693   7.91709    7.9194
##
## $coExpr[[2]]
##       GSM15785  GSM15790  GSM15787 GSM15791  GSM15788  GSM15792  GSM15795
## EGFR 1452.4300 1280.6000 1701.4000 1416.290 1551.1200 1384.5200 1779.4400
## GDF7   68.7106   59.7835   67.5011   58.521   62.3464   57.2744   69.3889
##       GSM15786  GSM15789  GSM15794
## EGFR 1379.5800 1724.8500 1130.1800
## GDF7   57.7585   67.9647   58.8681
##
## $coExpr[[3]]
##       GSM3911  GSM3913 GSM3909  GSM3912 GSM3914 GSM3910  GSM3918  GSM3915
## EGFR 264.1880 355.9830 265.477 471.5480 377.299 288.368 271.8510 263.7640
## GDF7  63.4734  28.6645 143.890  59.4679  15.271 126.904  77.6652  57.4021
##      GSM3916  GSM3919 GSM3920   GSM3917
## EGFR 126.383 1752.890 577.190 1759.9300
## GDF7 187.364  197.137  90.366   49.2624
##
##
## $corAnalysis
## $corAnalysis[[1]]
## $corAnalysis[[1]]$pearson
##   pearsoncor pearsonpval adjusted.pearsonpval
## 1   0.889057  0.01777975           0.05270126
##
## $corAnalysis[[1]]$spearman
##   spearmancor spearmanpval
## 1   0.7142857    0.1361111
##
## $corAnalysis[[1]]$kendall
##   kendallcor kendallpval
## 1  0.4666667   0.2722222
##
##
## $corAnalysis[[2]]
## $corAnalysis[[2]]$pearson
##   pearsoncor pearsonpval adjusted.pearsonpval
## 1   0.784558  0.00720065           0.02588214
##
## $corAnalysis[[2]]$spearman
##   spearmancor spearmanpval
## 1   0.7333333   0.02116648
##
## $corAnalysis[[2]]$kendall
##   kendallcor kendallpval
## 1  0.5555556  0.02860946
##
##
## $corAnalysis[[3]]
## $corAnalysis[[3]]$pearson
##   pearsoncor pearsonpval adjusted.pearsonpval
## 1  0.1628474   0.6130792            0.7266001
##
## $corAnalysis[[3]]$spearman
##   spearmancor spearmanpval
## 1  -0.1818182    0.5729582
##
## $corAnalysis[[3]]$kendall
##   kendallcor kendallpval
## 1 -0.1212121   0.6383613
##
```

```
##
##
## $Interactiontype
## [1] "None"
##
## $Coherency
## [1] "empty"
```

All the p-values and correlations are stored at these objects.

```
pearsonpvalues = c()
spearmanpvalues = c()
kendallpvalues = c()
pearsoncors = c()
spearmancors = c()
kendallcors = c()


for(i in 1:1000){
  if(unconnected.SignalingNet[[i]]$length > 1){
    for(j in 1:length(unconnected.SignalingNet[[i]]$corAnalysis)){

      a = unconnected.SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsonpval
      pearsonpvalues = c(pearsonpvalues,a)
      b =
unconnected.SignalingNet[[i]]$corAnalysis[[j]]$spearman$spearmanpval
      spearmanpvalues = c(spearmanpvalues,b)
      c = unconnected.SignalingNet[[i]]$corAnalysis[[j]]$kendall$kendallpval
      kendallpvalues = c(kendallpvalues,c)
      d = unconnected.SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor
      pearsoncors = c(pearsoncors,d)
      e = unconnected.SignalingNet[[i]]$corAnalysis[[j]]$spearman$spearmancor
      spearmancors = c(spearmancors,e)
      f = unconnected.SignalingNet[[i]]$corAnalysis[[j]]$kendall$kendallcor
      kendallcors = c(kendallcors,f)



    }
  }else {

    a = unconnected.SignalingNet[[i]]$corAnalysis$pearson$pearsonpval
    pearsonpvalues = c(pearsonpvalues,a)
    b = unconnected.SignalingNet[[i]]$corAnalysis$spearman$spearmanpval
    spearmanpvalues = c(spearmanpvalues,b)
    c = unconnected.SignalingNet[[i]]$corAnalysis$kendall$kendallpval
    kendallpvalues = c(kendallpvalues,c)
    d = unconnected.SignalingNet[[i]]$corAnalysis$pearson$pearsoncor
    pearsoncors = c(pearsoncors,d)
    e = unconnected.SignalingNet[[i]]$corAnalysis$spearman$spearmancor
    spearmancors = c(spearmancors,e)
    f = unconnected.SignalingNet[[i]]$corAnalysis$kendall$kendallcor
    kendallcors = c(kendallcors,f)
  }

  #print(i)
}
```

All the adjusted Pearson p-values are stored at the **adjusted.pearsonpvalues** object.

```
adjusted.pearsonpvalues = p.adjust(pearsonpvalues , method = "fdr")
adjusted.pearsonpvalues

all(lapply(unconnected.SignalingNet , function(x) x$length) > 0)
```

Adjusted Pearson p-values are added to the unconnected.SignalingNet object. After completing the process, adjusted.pearsonpvalues object should be empty or NULL.

```
for(i in 1:1000){
  l = unconnected.SignalingNet[[i]]$length
  if(l > 1){
    for(j in 1:l){

unconnected.SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval =
adjusted.pearsonpvalues[j]
    }
    adjusted.pearsonpvalues = adjusted.pearsonpvalues[-1:-l]
  }else{
    unconnected.SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval =
adjusted.pearsonpvalues[1]
    adjusted.pearsonpvalues = adjusted.pearsonpvalues[-1]
  }
}
```

```
adjusted.pearsonpvalues

## numeric(0)
```

The next codes are for getting the number of edges which are engaged in unconnected gene pairs (Table 2). The indices of gene pairs having p-values > 0.05 or p-values == NA are stored at **index.pval.cor.NA** object.

```
index.pval.cor.NA = c()

for(i in 1:1000){
  logic = c()
  if(unconnected.SignalingNet[[i]]$length > 1){

    for(j in 1:unconnected.SignalingNet[[i]]$length){


logic[j]=any(unlist(lapply(unconnected.SignalingNet[[i]]$corAnalysis[[j]],is.
na))) |
unconnected.SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval >
0.05
    }
    if(all(logic)){ index.pval.cor.NA = c(index.pval.cor.NA,i) }

  }else{
if(any(unlist(lapply(unconnected.SignalingNet[[i]]$corAnalysis,is.na)))|
unconnected.SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval >
0.05){
    index.pval.cor.NA = c(index.pval.cor.NA,i)}
  }
}

length(index.pval.cor.NA)

## [1] 437
```

The indices of gene pairs which have p-values < 0.05 and correlation > 0 are stored at **index.pval.cor1** object.

```
index.pval.cor1 = c()

for(i in 1:1000){
  logic =c()
  if(unconnected.SignalingNet[[i]]$length > 1){

    logic1=c()
    for(j in 1:unconnected.SignalingNet[[i]]$length){
      logic1[j] =
!any(unlist(lapply(unconnected.SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic1)){

      for(j in 1:unconnected.SignalingNet[[i]]$length){

        logic[j] =
unconnected.SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval <
0.05 & unconnected.SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
      }
      if(all(logic)) {
        index.pval.cor1 = c(index.pval.cor1,i)
      }
    }
  } else {
if(!any(unlist(lapply(unconnected.SignalingNet[[i]]$corAnalysis,is.na)))){
    a =
unconnected.SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05
& unconnected.SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
    if(a){
      index.pval.cor1 = c(index.pval.cor1,i)
    }
  }
  }

}

length(index.pval.cor1)

## [1] 59
```

The indices of gene pairs which have p-values < 0.05 and correlation < 0 are stored at **index.pval.cor1** object.

```
index.pval.cor2 = c()

for(i in 1:1000){
  logic =c()
  if(unconnected.SignalingNet[[i]]$length > 1){

    logic1=c()
    for(j in 1:unconnected.SignalingNet[[i]]$length){
      logic1[j] =
!any(unlist(lapply(unconnected.SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic1)){

      for(j in 1:unconnected.SignalingNet[[i]]$length){

        logic[j] =
unconnected.SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval <
0.05 & unconnected.SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic)) {
        index.pval.cor2 = c(index.pval.cor2,i)
      }
    }
  } else {
if(!any(unlist(lapply(unconnected.SignalingNet[[i]]$corAnalysis,is.na)))){
    a =
unconnected.SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05
& unconnected.SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
    if(a){
      index.pval.cor2 = c(index.pval.cor2,i)
    }
  }
  }

}

length(index.pval.cor2)

## [1] 33
```

```
1000 - 392 - 52 - 51

## [1] 505
```

7.Two Edge Subgraphs

```
load("Two Edge Subgraphs KEGG GEO.RData")
```

The following code shows if there is any loop among the eligible edges.

```
logic=c()
for(i in Eligibles){
  logic[i] = edgelist[i,2]==edgelist[i,3]
}
idxloop = which(logic)
edgelist[idxloop,]

## [1] ID               Gene1            Gene2            Interaction type
## <0 rows> (or 0-length row.names)

# There is no loop
```

Edges which has been participated in two-edge subgraphs are saved at **twoEdges** edgelist. the indices of these edges are stored at **idxTWOedges** object.

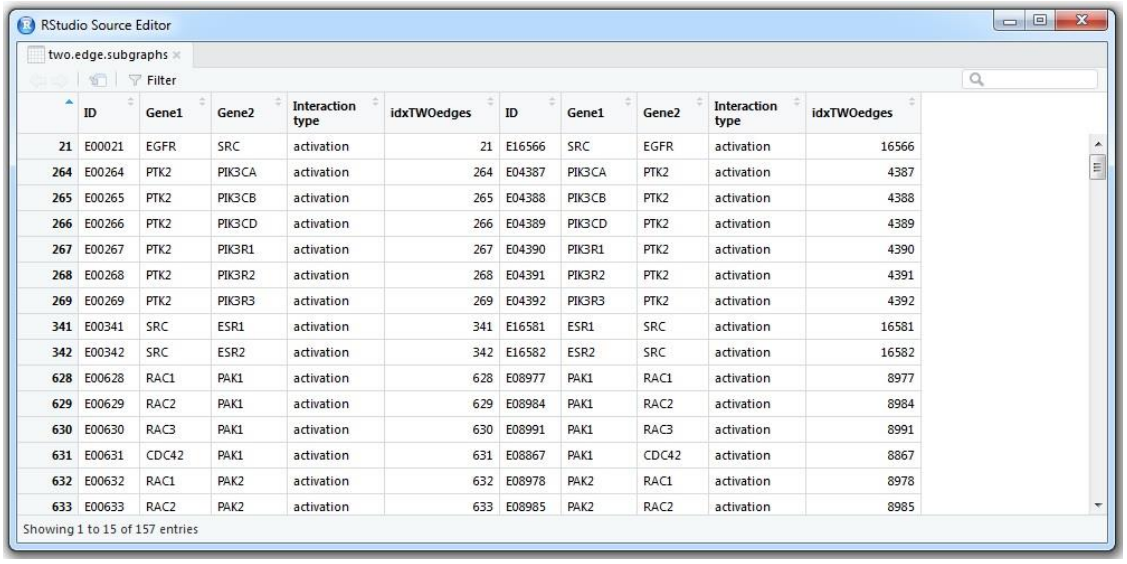
```
idxTWOedges = list()
for(i in Eligibles){
  a = edgelist[i,2]
  b = edgelist[i,3]

  index = which(edgelist[,2] == b & edgelist[,3] == a)
  if(length(index) > 0){idxTWOedges[[i]] = c(i,index)}

}
idxTWOedges = unlist(idxTWOedges)

length(idxTWOedges)
twoEdges=edgelist[idxTWOedges,]
sum(duplicated(twoEdges))
sum(duplicated(edgelist[2:4]))

twoEdges = twoEdges[!duplicated(twoEdges),]
idxTWOedges = rownames(twoEdges)
idxTWOedges = as.numeric(idxTWOedges)
```

```
length(idxTWOedges)

## [1] 314

tail(twoEdges)

##            ID  Gene1  Gene2 Interaction type idxTWOedges
## 20272 E20272 CDKN1B  CCND2       inhibition       20272
## 25769 E25769  CCND2 CDKN1B       inhibition       25769
## 20273 E20273 CDKN1B  CCND3       inhibition       20273
## 25770 E25770  CCND3 CDKN1B       inhibition       25770
## 21710 E21710 CSNK1E  WWTR1       inhibition       21710
## 21726 E21726  WWTR1 CSNK1E       inhibition       21726
```

```
head(twoEdges)

##            ID  Gene1  Gene2 Interaction type idxTWOedges
## 21    E00021   EGFR    SRC       activation          21
## 16566 E16566    SRC   EGFR       activation       16566
## 264   E00264   PTK2 PIK3CA       activation         264
## 4387  E04387 PIK3CA   PTK2       activation        4387
## 265   E00265   PTK2 PIK3CB       activation         265
## 4388  E04388 PIK3CB   PTK2       activation        4388
```

Two-edge subgraphs are stored at **two.edge.subgraphs** object. This table is a **bisection edgelist**, and it contains columns which determine the indices of edges that has been participated in two edge subgraphs. Each row of the table presents a two-edge subgraph (Figure8).

```
twoEdges$idxTWOedges = idxTWOedges

which(duplicated(edgelist[,2:4]))

index = seq(1 , length(idxTWOedges) , by = 2 )

two.edge.subgraphs = cbind(twoEdges[index,] , twoEdges[index + 1 ,])
```



	ID	Gene1	Gene2	Interaction type	idxTWOedges	ID	Gene1	Gene2	Interaction type	idxTWOedges
21	E00021	EGFR	SRC	activation	21	E16566	SRC	EGFR	activation	16566
264	E00264	PTK2	PIK3CA	activation	264	E04387	PIK3CA	PTK2	activation	4387
265	E00265	PTK2	PIK3CB	activation	265	E04388	PIK3CB	PTK2	activation	4388
266	E00266	PTK2	PIK3CD	activation	266	E04389	PIK3CD	PTK2	activation	4389
267	E00267	PTK2	PIK3R1	activation	267	E04390	PIK3R1	PTK2	activation	4390
268	E00268	PTK2	PIK3R2	activation	268	E04391	PIK3R2	PTK2	activation	4391
269	E00269	PTK2	PIK3R3	activation	269	E04392	PIK3R3	PTK2	activation	4392
341	E00341	SRC	ESR1	activation	341	E16581	ESR1	SRC	activation	16581
342	E00342	SRC	ESR2	activation	342	E16582	ESR2	SRC	activation	16582
628	E00628	RAC1	PAK1	activation	628	E08977	PAK1	RAC1	activation	8977
629	E00629	RAC2	PAK1	activation	629	E08984	PAK1	RAC2	activation	8984
630	E00630	RAC3	PAK1	activation	630	E08991	PAK1	RAC3	activation	8991
631	E00631	CDC42	PAK1	activation	631	E08867	PAK1	CDC42	activation	8867
632	E00632	RAC1	PAK2	activation	632	E08978	PAK2	RAC1	activation	8978
633	E00633	RAC2	PAK2	activation	633	E08985	PAK2	RAC2	activation	8985

Showing 1 to 15 of 157 entries

Figure8: The bisection edgelist

DNFBL-two-edge subgrphs are stored at the bisection edgelist called **DNFBL**, DPFBL1-two-edge subgrphs are stored at the bisection edgellist called **DPFBL1** and DPFBL2-two-edge subgrphs are stored at the bisection edgellist called **DPFBL2** (Tables 1 and 2).

```
idxDNFBL.1 = which(two.edge.subgraphs[,4]=="activation" &
two.edge.subgraphs[,9]=="inhibition")
DNFBL.1 = two.edge.subgraphs[idxDNFBL.1,]

length(DNFBL.1[,1])

## [1] 20

head(DNFBL.1)

##          ID   Gene1     Gene2 Interaction type idxTWOedges     ID
Gene1
## 1741 E01741    RAC1      RHOA       activation        1741 E25247
RHOA
## 1742 E01742    RAC2      RHOA       activation        1742 E25248
RHOA
## 1743 E01743    RAC3      RHOA       activation        1743 E25249
RHOA
## 2900 E02900  PRKACB   PPP1R1B       activation        2900 E25642
PPP1R1B
## 3107 E03107 TNFSF11 TNFRSF11B       activation        3107 E21675
TNFRSF11B
## 4885 E04885    INSR    PTPN11       activation        4885 E25571
PTPN11
##         Gene2 Interaction type idxTWOedges
## 1741     RAC1       inhibition       25247
## 1742     RAC2       inhibition       25248
## 1743     RAC3       inhibition       25249
## 2900   PRKACB       inhibition       25642
## 3107  TNFSF11       inhibition       21675
## 4885     INSR       inhibition       25571

idxDNFBL.2 = which(two.edge.subgraphs[,4]=="inhibition" &
two.edge.subgraphs[,9]=="activation")
DNFBL.2 = two.edge.subgraphs[idxDNFBL.2,]

DNFBL = rbind(DNFBL.1 , DNFBL.2)
idxDNFBL = c(idxDNFBL.1,idxDNFBL.2)
```

```
idxDPFBL1 = which(two.edge.subgraphs[,4]=="activation" &
two.edge.subgraphs[,9]=="activation")
DPFBL1 = two.edge.subgraphs[idxDPFBL1,]

length(DPFBL1[,1])

## [1] 125

head(DPFBL1)
```

```
##          ID Gene1  Gene2 Interaction type idxTWOedges     ID  Gene1 Gene2
## 21  E00021  EGFR    SRC       activation          21 E16566    SRC  EGFR
## 264 E00264  PTK2 PIK3CA       activation         264 E04387 PIK3CA  PTK2
## 265 E00265  PTK2 PIK3CB       activation         265 E04388 PIK3CB  PTK2
## 266 E00266  PTK2 PIK3CD       activation         266 E04389 PIK3CD  PTK2
## 267 E00267  PTK2 PIK3R1       activation         267 E04390 PIK3R1  PTK2
## 268 E00268  PTK2 PIK3R2       activation         268 E04391 PIK3R2  PTK2
##     Interaction type idxTWOedges
## 21        activation       16566
## 264       activation        4387
## 265       activation        4388
## 266       activation        4389
## 267       activation        4390
## 268       activation        4391
```

```
idxDPFBL2 = which(two.edge.subgraphs[,4]=="inhibition" &
two.edge.subgraphs[,9]=="inhibition")
DPFBL2 = two.edge.subgraphs[idxDPFBL2,]

length(DPFBL2[,1])

## [1] 12

head(DPFBL2)

##            ID  Gene1  Gene2 Interaction type idxTWOedges     ID  Gene1
Gene2
## 21354 E21354   GLI1   GLI3       inhibition       21354 E21350   GLI3
GLI1
## 19192 E19192   BCL2    BAX       inhibition       19192 E26437    BAX
BCL2
## 19202 E19202   BCL2    BAD       inhibition       19202 E19216    BAD
BCL2
## 19217 E19217 BCL2L1   BAK1       inhibition       19217 E25805   BAK1
BCL2L1
## 19841 E19841   CDK2 CDKN1B       inhibition       19841 E19850 CDKN1B
CDK2
## 19842 E19842   CDK2 CDKN1C       inhibition       19842 E19851 CDKN1C
CDK2
##       Interaction type idxTWOedges
## 21354       inhibition       21350
## 19192       inhibition       26437
## 19202       inhibition       19216
## 19217       inhibition       25805
## 19841       inhibition       19850
## 19842       inhibition       19851
```

7.1.Computing the number of edges constructing in three kinds of two-edge subgraphs

DNFBL

```r
index = c(DNFBL[,5],DNFBL[,10])


DNFBL.pval.cor1 = 0
DNFBL.pval.cor.NA = 0
DNFBL.pval.cor2 = 0




for(i in index){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {DNFBL.pval.cor1=DNFBL.pval.cor1+1}
      if(all(logic2)) {DNFBL.pval.cor2=DNFBL.pval.cor2+1}
    }
  } else{
    if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){DNFBL.pval.cor1=DNFBL.pval.cor1+1}
      if(b){DNFBL.pval.cor2=DNFBL.pval.cor2+1}
    }
  }
}




for(i in index){
  logic = c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){

      logic[j]=any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
| SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
    }
    if(all(logic)){ DNFBL.pval.cor.NA = DNFBL.pval.cor.NA + 1 }

  }else{ if(any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))|
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05){
    index.pval.cor.NA = DNFBL.pval.cor.NA + 1}
  }
}

DNFBL.pval.cor1

## [1] 0

DNFBL.pval.cor.NA

## [1] 10

DNFBL.pval.cor2

## [1] 3

length(DNFBL[,1]) * 2 - c(DNFBL.pval.cor1 + DNFBL.pval.cor.NA +
DNFBL.pval.cor2)

## [1] 27
```

DPFBL1

```
index = c(DPFBL1[,5] , DPFBL1[,10])



DPFBL1.pval.cor1 = 0
DPFBL1.pval.cor.NA = 0
DPFBL1.pval.cor2 = 0
```

```r
for(i in index){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=(!(class(SignalingNet[[i]]$corAnalysis[[j]]) == "logical" |
any(is.na(unlist(SignalingNet[[i]]$corAnalysis[[j]])))))
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {DPFBL1.pval.cor1=DPFBL1.pval.cor1+1}
      if(all(logic2)) {DPFBL1.pval.cor2=DPFBL1.pval.cor2+1}
    }
  } else if(SignalingNet[[i]]$length == 1) {
    if(!(class(SignalingNet[[i]]$corAnalysis) == "logical" |
any(is.na(unlist(SignalingNet[[i]]$corAnalysis))))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){DPFBL1.pval.cor1=DPFBL1.pval.cor1+1}
      if(b){DPFBL1.pval.cor2=DPFBL1.pval.cor2+1}
    }
  }
}




for(i in index){
  logic =c()
  if(SignalingNet[[i]]$length > 1){
    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=!class(SignalingNet[[i]]$corAnalysis[[j]]) == "logical"
    }
    if(all(logic)){
      logic1 =c()
      for(j in 1:SignalingNet[[i]]$length){
        logic1[j]=
is.na(SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval) |
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
      }
      if(all(logic1)){DPFBL1.pval.cor.NA = DPFBL1.pval.cor.NA+1}
    }

  } else {
    if(!class(SignalingNet[[i]]$corAnalysis) == "logical"){
      a = is.na(SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval) |
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05
      if(a){DPFBL1.pval.cor.NA=DPFBL1.pval.cor.NA+1}
    }
  }
}

DPFBL1.pval.cor1

## [1] 34

DPFBL1.pval.cor.NA

## [1] 85

DPFBL1.pval.cor2

## [1] 6

length(DPFBL1[,1]) * 2 - c(DPFBL1.pval.cor1 + DPFBL1.pval.cor.NA +
DPFBL1.pval.cor2)

## [1] 125
```

DPFBL2

```
index = c(DPFBL2[,5] , DPFBL2[,10])


DPFBL2.pval.cor1 = 0
DPFBL2.pval.cor.NA = 0
DPFBL2.pval.cor2 = 0



for(i in index){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=(!(class(SignalingNet[[i]]$corAnalysis[[j]]) == "logical" |
any(is.na(unlist(SignalingNet[[i]]$corAnalysis[[j]])))))
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {DPFBL2.pval.cor1=DPFBL2.pval.cor1+1}
      if(all(logic2)) {DPFBL2.pval.cor2=DPFBL2.pval.cor2+1}
    }
  } else if(SignalingNet[[i]]$length == 1) {
    if(!(class(SignalingNet[[i]]$corAnalysis) == "logical" |
any(is.na(unlist(SignalingNet[[i]]$corAnalysis))))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){DPFBL2.pval.cor1=DPFBL2.pval.cor1+1}
      if(b){DPFBL2.pval.cor2=DPFBL2.pval.cor2+1}
    }
  }
}




for(i in index){
  logic =c()
  if(SignalingNet[[i]]$length > 1){
    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=!class(SignalingNet[[i]]$corAnalysis[[j]]) == "logical"
    }
    if(all(logic)){
      logic1 =c()
      for(j in 1:SignalingNet[[i]]$length){
        logic1[j]=
is.na(SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval) |
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
      }
      if(all(logic1)){DPFBL2.pval.cor.NA = DPFBL2.pval.cor.NA+1}
    }

  } else {
    if(!class(SignalingNet[[i]]$corAnalysis) == "logical"){
      a = is.na(SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval) |
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05
      if(a){DPFBL2.pval.cor.NA=DPFBL2.pval.cor.NA+1}
    }
  }
}

DPFBL2.pval.cor1

## [1] 0

DPFBL2.pval.cor.NA

## [1] 10

DPFBL2.pval.cor2

## [1] 0

length(DPFBL2[,1]) * 2 - (DPFBL2.pval.cor1 + DPFBL2.pval.cor.NA +
DPFBL2.pval.cor2)

## [1] 14
```

8.Multiple Edge subgraphs

To compute the number of edges which participate in multiple-edge subgraphs, an edgelist is created from eligible edges called **shortEdgelist**.

```
library(igraph)

shortEdgelist = edgelist[Eligibles,]

shortEdgelist[which(shortEdgelist[,4] == "activation"),4] = 1
shortEdgelist[which(shortEdgelist[,4] == "inhibition"),4] = -1
```

A weighted graph was created from shortEdgelist object, and -1 was assigned to inhibitions and 1 was assigned to activations as weights. Then, the largest component of the eligibile edges in the graph was extracted called **weighted.giant.component**. We changed this object into an adjacency matrix called **WadjaMatrix**. The same processes were done for non-weighted adjacency matrix and an object called **none.weighted.giant.component** was created. Then, **AdjaMatrix** object was created from that object. The diameter of the two giant components was computed and using matrix self-multiplication, the number of edges which are engaged in 8 kinds of multiple-edge subgraphs was computed (Table1 and Table2).

If the edge i in the edgelist is activation and the entity between the gene pair in one of the weighted matrices equals to their unweighted counterparts while the entities are not zero or NA, this edge is engaged in MFFL1 or MPFBL1 subgraphs.

If the edge i in the edgelist is inhibition and the entity between the gene pair in one of the weighted matrices equals to their unweighted counterparts while the entities are not zero or NA, this edge is engaged in MNFFL2 or MNFBL2 subgraphs.

If the edge i in the edgelist is activation and the entity between the gene pair in one of the weighted matrices does not equal to their unweighted counterparts while the entities are not zero or NA, this edge is engaged in MNFBL1 or MNFFL1 subgraphs.

If the edge i in the edgelist is inhibition and the entity between the gene pair in one of the weighted matrices does not equal to their unweighted counterparts while the entities are not zero or NA, this edge is engaged in MFFL2 or MPFBL2 subgraphs.

```
w = as.numeric(shortEdgelist[,4])
shortEdgelist = as.matrix(shortEdgelist)

g = graph_from_edgelist(shortEdgelist[,2:3] , directed = T)
E(g)$weights = w
```

```
shortEdgelistGenes = unique(c(shortEdgelist[,2] , c(shortEdgelist[,3])))

c= components(g)
```

```
weighted.giant.component = induced.subgraph(g , which(c$membership ==  1))

weighted.giant.component

## IGRAPH e173162 DN-- 2549 15718 --
## + attr: name (v/c), weights (e/n)
## + edges from e173162 (vertex names):
##  [1] EGF  ->EGFR   TGFA ->EGFR   HGF  ->MET    MET  ->ERBB3  IGF1 ->IGF1R
##  [6] VEGFA->KDR    PDGFA->PDGFRA PDGFB->PDGFRA PDGFC->PDGFRA PDGFD->PDGFRA
## [11] PDGFA->PDGFRB PDGFB->PDGFRB PDGFC->PDGFRB PDGFD->PDGFRB FGF2 ->FGFR3
## [16] FGF2 ->FGFR2  GAS6 ->AXL    IL6  ->IL6R   EGFR ->JAK1   EGFR ->JAK2
## [21] EGFR ->SRC    EGFR ->GAB1   EGFR ->PLCG1  EGFR ->PLCG2  EGFR ->SHC2
## [26] EGFR ->SHC4   EGFR ->SHC3   EGFR ->SHC1   MET  ->JAK1   MET  ->JAK2
## [31] MET  ->SRC    MET  ->GAB1   MET  ->PLCG1  MET  ->PLCG2  MET  ->SHC2
## [36] MET  ->SHC4   MET  ->SHC3   MET  ->SHC1   IGF1R->JAK1   IGF1R->JAK2
## + ... omitted several edges

is.connected(weighted.giant.component)

## [1] TRUE

WadjMatrix = as_adjacency_matrix(weighted.giant.component , attr = "weights")

WadjMatrix = as.matrix(WadjMatrix)

diameter(weighted.giant.component)

## [1] 17
```

```
Wmatpower2 = WadjMatrix  %*% WadjMatrix

Wmatpower3 = Wmatpower2  %*% WadjMatrix

Wmatpower4 = Wmatpower3  %*% WadjMatrix

Wmatpower5 = Wmatpower4  %*% WadjMatrix

Wmatpower6 = Wmatpower5 %*% WadjMatrix

Wmatpower7 = Wmatpower6 %*% WadjMatrix

Wmatpower8 = Wmatpower7 %*% WadjMatrix

Wmatpower9 = Wmatpower8  %*% WadjMatrix

Wmatpower10 = Wmatpower9  %*% WadjMatrix
```

```
Wmatpower11 = Wmatpower10  %*% WadjMatrix

Wmatpower12 = Wmatpower11  %*% WadjMatrix

Wmatpower13 = Wmatpower12 %*% WadjMatrix

Wmatpower14 = Wmatpower13 %*% WadjMatrix

Wmatpower15 = Wmatpower14  %*% WadjMatrix

Wmatpower16 = Wmatpower15  %*% WadjMatrix

Wmatpower17 = Wmatpower16  %*% WadjMatrix
```

```
g1 = graph_from_edgelist(shortEdgelist[,2:3] , directed = T)

c= components(g1)

none.weighted.giant.component = induced.subgraph(g1 , which(c$membership ==
1))

none.weighted.giant.component

## IGRAPH 4d382e2 DN-- 2549 15718 --
## + attr: name (v/c)
## + edges from 4d382e2 (vertex names):
##  [1] EGF  ->EGFR   TGFA ->EGFR   HGF  ->MET    MET  ->ERBB3  IGF1 ->IGF1R
##  [6] VEGFA->KDR    PDGFA->PDGFRA PDGFB->PDGFRA PDGFC->PDGFRA PDGFD->PDGFRA
## [11] PDGFA->PDGFRB PDGFB->PDGFRB PDGFC->PDGFRB PDGFD->PDGFRB FGF2 ->FGFR3
## [16] FGF2 ->FGFR2  GAS6 ->AXL    IL6  ->IL6R   EGFR ->JAK1   EGFR ->JAK2
## [21] EGFR ->SRC    EGFR ->GAB1   EGFR ->PLCG1  EGFR ->PLCG2  EGFR ->SHC2
## [26] EGFR ->SHC4   EGFR ->SHC3   EGFR ->SHC1   MET  ->JAK1   MET  ->JAK2
## [31] MET  ->SRC    MET  ->GAB1   MET  ->PLCG1  MET  ->PLCG2  MET  ->SHC2
## [36] MET  ->SHC4   MET  ->SHC3   MET  ->SHC1   IGF1R->JAK1   IGF1R->JAK2
## + ... omitted several edges

is.connected(none.weighted.giant.component)

## [1] TRUE

AdjMatrix = as_adjacency_matrix(none.weighted.giant.component)

AdjMatrix = as.matrix(AdjMatrix)

diameter(none.weighted.giant.component)

## [1] 17
```

```
matpower2 = AdjMatrix  %*% AdjMatrix

matpower3 = matpower2  %*% AdjMatrix

matpower4 = matpower3  %*% AdjMatrix

matpower5 = matpower4  %*% AdjMatrix

matpower6 = matpower5 %*% AdjMatrix

matpower7 = matpower6 %*% AdjMatrix

matpower8 = matpower7 %*% AdjMatrix

matpower9 = matpower8  %*% AdjMatrix

matpower10 = matpower9  %*% AdjMatrix

matpower11 = matpower10  %*% AdjMatrix

matpower12 = matpower11  %*% AdjMatrix

matpower13 = matpower12 %*% AdjMatrix

matpower14 = matpower13 %*% AdjMatrix

matpower15 = matpower14  %*% AdjMatrix

matpower16 = matpower15  %*% AdjMatrix

matpower17 = matpower16  %*% AdjMatrix
```

listAdj1 object is a large list containing matrices as the same dimension as the adjacency matrices. This list contains 16 matrices, and Each matrix is representative of one of the powered matrices.

```
Apower = matrix(0 , 2549 , 2549)
colnames(Apower) = colnames(AdjMatrix)
rownames(Apower) = rownames(AdjMatrix)
listAdj1 = list(Apower2 = Apower,
Apower3 = Apower,
Apower4 = Apower,
Apower5 = Apower,
Apower6 = Apower,
Apower7 = Apower,
Apower8 = Apower,
Apower9 = Apower,
Apower10 = Apower,
Apower11 = Apower,
Apower12 = Apower,
Apower13 = Apower,
Apower14 = Apower,
Apower15 = Apower,
Apower16 = Apower,
Apower17 = Apower
)
```

Before computing the number of edges which participate in multiple-edge subgraphs, the following algorithms are required to be applied: If WmatpowerX[i,j] == matpowerX[i,j], we put the value of WmatpowerX[i,j] at the ApowerX[i,j] in **listAdj1** list. Otherwise, we put NA. After that, the number of edges which take part in "MPFBL1", "MNFBL2", "MFFL1" and "MNFFL2" multiple-edge subgraphs are computed. For the other multiple-edge subgraphs called "MNFBL1", "MPFBL2", "MFFL2" and "MNFFL1", If WmatpowerX[i,j] != matpowerX[i,j], we put the value of WmatpowerX[i,j] at the ApowerX[i,j] in **listAdj2** list. Otherwise, we put NA.

```
for(i in 1:length(AdjMatrix[,1])){

for(j in 1:length(AdjMatrix[1,])){

if(Wmatpower2[i,j] == matpower2[i,j]){
  listAdj1$Apower2[i,j] = Wmatpower2[i,j]
} else { listAdj1$Apower2[i,j] = NA }


if(Wmatpower3[i,j] == matpower3[i,j]){
  listAdj1$Apower3[i,j] = Wmatpower3[i,j]
} else { listAdj1$Apower3[i,j] = NA }


if(Wmatpower4[i,j] == matpower4[i,j]){
  listAdj1$Apower4[i,j] = Wmatpower4[i,j]
} else { listAdj1$Apower4[i,j] = NA }


if(Wmatpower5[i,j] == matpower5[i,j]){
  listAdj1$Apower5[i,j] = Wmatpower5[i,j]
} else { listAdj1$Apower5[i,j] = NA }


if(Wmatpower6[i,j] == matpower6[i,j]){
```

```r
  listAdj1$Apower6[i,j] = Wmatpower6[i,j]
} else { listAdj1$Apower6[i,j] = NA }


if(Wmatpower7[i,j] == matpower7[i,j]){
  listAdj1$Apower7[i,j] = Wmatpower7[i,j]
} else { listAdj1$Apower7[i,j] = NA }


if(Wmatpower8[i,j] == matpower8[i,j]){
  listAdj1$Apower8[i,j] = Wmatpower8[i,j]
} else { listAdj1$Apower8[i,j] = NA }


if(Wmatpower9[i,j] == matpower9[i,j]){
  listAdj1$Apower9[i,j] = Wmatpower9[i,j]
} else { listAdj1$Apower9[i,j] = NA }


if(Wmatpower10[i,j] == matpower10[i,j]){
  listAdj1$Apower10[i,j] = Wmatpower10[i,j]
} else { listAdj1$Apower10[i,j] = NA }


if(Wmatpower11[i,j] == matpower11[i,j]){
  listAdj1$Apower11[i,j] = Wmatpower11[i,j]
} else { listAdj1$Apower11[i,j] = NA }


if(Wmatpower12[i,j] == matpower12[i,j]){
  listAdj1$Apower12[i,j] = Wmatpower12[i,j]
} else { listAdj1$Apower12[i,j] = NA }


if(Wmatpower13[i,j] == matpower13[i,j]){
  listAdj1$Apower13[i,j] = Wmatpower13[i,j]
} else { listAdj1$Apower13[i,j] = NA }


if(Wmatpower14[i,j] == matpower14[i,j]){
  listAdj1$Apower14[i,j] = Wmatpower14[i,j]
} else { listAdj1$Apower14[i,j] = NA }


if(Wmatpower15[i,j] == matpower15[i,j]){
  listAdj1$Apower15[i,j] = Wmatpower15[i,j]
} else { listAdj1$Apower15[i,j] = NA }
```

```r
if(Wmatpower16[i,j] == matpower16[i,j]){
  listAdj1$Apower16[i,j] = Wmatpower16[i,j]
} else { listAdj1$Apower16[i,j] = NA }


if(Wmatpower17[i,j] == matpower17[i,j]){
  listAdj1$Apower17[i,j] = Wmatpower17[i,j]
} else { listAdj1$Apower17[i,j] = NA }

}
  # print(i)
}
```

```r
Apower = matrix(0 , 2549 , 2549)
colnames(Apower) = colnames(AdjMatrix)
rownames(Apower) = rownames(AdjMatrix)
listAdj2 = list(Apower2 = Apower,
               Apower3 = Apower,
               Apower4 = Apower,
               Apower5 = Apower,
               Apower6 = Apower,
               Apower7 = Apower,
               Apower8 = Apower,
               Apower9 = Apower,
               Apower10 = Apower,
               Apower11 = Apower,
               Apower12 = Apower,
               Apower13 = Apower,
               Apower14 = Apower,
               Apower15 = Apower,
               Apower16 = Apower,
               Apower17 = Apower
)

for(i in 1:length(AdjMatrix[,1])){

  for(j in 1:length(AdjMatrix[1,])){

    if(Wmatpower2[i,j] != matpower2[i,j]){
      listAdj2$Apower2[i,j] = Wmatpower2[i,j]
    } else { listAdj2$Apower2[i,j] = NA }


    if(Wmatpower3[i,j] != matpower3[i,j]){
      listAdj2$Apower3[i,j] = Wmatpower3[i,j]
    } else { listAdj2$Apower3[i,j] = NA }
    if(Wmatpower4[i,j] != matpower4[i,j]){
      listAdj2$Apower4[i,j] = Wmatpower4[i,j]
    } else { listAdj2$Apower4[i,j] = NA }


    if(Wmatpower5[i,j] != matpower5[i,j]){
      listAdj2$Apower5[i,j] = Wmatpower5[i,j]
    } else { listAdj2$Apower5[i,j] = NA }


    if(Wmatpower6[i,j] != matpower6[i,j]){
      listAdj2$Apower6[i,j] = Wmatpower6[i,j]
    } else { listAdj2$Apower6[i,j] = NA }


    if(Wmatpower7[i,j] != matpower7[i,j]){
      listAdj2$Apower7[i,j] = Wmatpower7[i,j]
    } else { listAdj2$Apower7[i,j] = NA }


    if(Wmatpower8[i,j] != matpower8[i,j]){
      listAdj2$Apower8[i,j] = Wmatpower8[i,j]
    } else { listAdj2$Apower8[i,j] = NA }


    if(Wmatpower9[i,j] != matpower9[i,j]){
      listAdj2$Apower9[i,j] = Wmatpower9[i,j]
    } else { listAdj2$Apower9[i,j] = NA }


    if(Wmatpower10[i,j] != matpower10[i,j]){
      listAdj2$Apower10[i,j] = Wmatpower10[i,j]
    } else { listAdj2$Apower10[i,j] = NA }


    if(Wmatpower11[i,j] != matpower11[i,j]){
      listAdj2$Apower11[i,j] = Wmatpower11[i,j]
    } else { listAdj2$Apower11[i,j] = NA }


    if(Wmatpower12[i,j] != matpower12[i,j]){
      listAdj2$Apower12[i,j] = Wmatpower12[i,j]
    } else { listAdj2$Apower12[i,j] = NA }


    if(Wmatpower13[i,j] != matpower13[i,j]){
      listAdj2$Apower13[i,j] = Wmatpower13[i,j]
```

```r
    } else { listAdj2$Apower13[i,j] = NA }


    if(Wmatpower14[i,j] != matpower14[i,j]){
      listAdj2$Apower14[i,j] = Wmatpower14[i,j]
    } else { listAdj2$Apower14[i,j] = NA }


    if(Wmatpower15[i,j] != matpower15[i,j]){
      listAdj2$Apower15[i,j] = Wmatpower15[i,j]
    } else { listAdj2$Apower15[i,j] = NA }


    if(Wmatpower16[i,j] != matpower16[i,j]){
      listAdj2$Apower16[i,j] = Wmatpower16[i,j]
    } else { listAdj2$Apower16[i,j] = NA }


    if(Wmatpower17[i,j] != matpower17[i,j]){
      listAdj2$Apower17[i,j] = Wmatpower17[i,j]
    } else { listAdj2$Apower17[i,j] = NA }

  }

}
```

Using the following code, the indices of the giant component edges in the **KEGG edgelist** are found, and they are stored at **commonIndex** object.

```r
m = as_edgelist(weighted.giant.component)
m = as.data.frame(m , stringsAsFactors = F)
e = edgelist[,2:3]


index.of.rows.in.x.that.are.in.y  <- function(x,y)
{
  x.vec <- apply(x, 1, paste, collapse = "")
  y.vec <- apply(y, 1, paste, collapse = "")
  index = x.vec %in% y.vec
  return(which(index))
}
commonIndex = index.of.rows.in.x.that.are.in.y(e,m)
```

8.1.Computing the number of edges participating in MPFBL1, MNFBL2, MFFL1 and MNFFL2 multiple-edge subgraphs

The indices of edges taking part in MFFL1 subgraph are saved at **idxMFFL1** object through the following code.

```
MFFL1 = list()

idxMFFL1 = list()

for(i in commonIndex){

if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower2[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower2[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower2[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower3[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower3[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower3[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower4[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower4[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower4[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower5[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower5[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower5[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower6[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower6[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower6[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower7[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower7[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower7[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower8[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower8[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower8[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower9[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower9[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower9[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower10[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower10[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower10[edgelist[i,2],edgelist[i,3]]
```

```
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower11[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower11[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower11[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower12[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower12[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower12[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower13[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower13[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower13[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower14[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower14[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower14[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower15[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower15[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower15[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower16[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower16[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower16[edgelist[i,2],edgelist[i,3]]
} else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower17[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower17[edgelist[i,2],edgelist[i,3]] == 0)){
  MFFL1[i]=listAdj1$Apower17[edgelist[i,2],edgelist[i,3]]
}

if( (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower2[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower2[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower3[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower3[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower4[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower4[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower5[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower5[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower6[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower6[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower7[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower7[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower8[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower8[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower9[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower9[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower10[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower10[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower11[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower11[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower12[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower12[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower13[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower13[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower14[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower14[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower15[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower15[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower16[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower16[edgelist[i,2],edgelist[i,3]] == 0)) |
    (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower17[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower17[edgelist[i,2],edgelist[i,3]] == 0))
){idxMFFL1[i] = i}

}

MFFL1 = unlist(MFFL1)

idxMFFL1 = unlist(idxMFFL1)

length(idxMFFL1)

## [1] 8416
```

The indices of edges taking part in MPFBL1 subgraph are saved at **idxMPFBL1** object through the following code.

```
MPFBL1 = list()

idxMPFBL1 = list()
```

```
for(i in commonIndex){

  if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower2[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower2[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower2[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower3[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower3[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower3[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower4[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower4[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower4[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower5[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower5[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower5[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower6[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower6[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower6[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower7[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower7[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower7[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower8[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower8[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower8[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower9[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower9[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower9[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower10[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower10[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower10[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower11[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower11[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower11[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower12[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower12[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower12[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower13[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower13[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower13[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower14[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower14[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower14[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower15[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower15[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower15[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower16[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower16[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower16[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower17[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower17[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL1[i]=listAdj1$Apower17[edgelist[i,3],edgelist[i,2]]
  }

  if( (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower2[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower2[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower3[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower3[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower4[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower4[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower5[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower5[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower6[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower6[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower7[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower7[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower8[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower8[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower9[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower9[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower10[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower10[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower11[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower11[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower12[edgelist[i,3],edgelist[i,2]]) |
```

```r
listAdj1$Apower12[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower13[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower13[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower14[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower14[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower15[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower15[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower16[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower16[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj1$Apower17[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower17[edgelist[i,3],edgelist[i,2]] == 0))
  ){idxMPFBL1[i] = i}


}


MPFBL1 = unlist(MPFBL1)

idxMPFBL1 = unlist(idxMPFBL1)
length(idxMPFBL1)
```

```r
length(idxMPFBL1)
```

```
## [1] 2818
```

The indices of edges taking part in MNFFL2 subgraph are stored at **idxMNFFL2** object through the following code.

```r
MNFFL2 = list()

idxMNFFL2 = list()

for(i in commonIndex){

  if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower2[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower2[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower2[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower3[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower3[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower3[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower4[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower4[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower4[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower5[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower5[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower5[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower6[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower6[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower6[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower7[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower7[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower7[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower8[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower8[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower8[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower9[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower9[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower9[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower10[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower10[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower10[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower11[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower11[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower11[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower12[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower12[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower12[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower13[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower13[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower13[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower14[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower14[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower14[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower15[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower15[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower15[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower16[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower16[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower16[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower17[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower17[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFFL2[i]=listAdj1$Apower17[edgelist[i,2],edgelist[i,3]]
  }

  if( (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower2[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower2[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower3[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower3[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower4[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower4[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower5[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower5[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower6[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower6[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower7[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower7[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower8[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower8[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower9[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower9[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower10[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower10[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower11[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower11[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower12[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower12[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower13[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower13[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower14[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower14[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower15[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower15[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower16[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower16[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower17[edgelist[i,2],edgelist[i,3]]) |
listAdj1$Apower17[edgelist[i,2],edgelist[i,3]] == 0))
  ){idxMNFFL2[i] = i}


}

MNFFL2 = unlist(MNFFL2)

idxMNFFL2 = unlist(idxMNFFL2)

length(idxMNFFL2)

## [1] 659
```

The indices of edges taking part in MNFBL2 subgraph are stored at **idxMNFBL2** object through the following code.

```
MNFBL2 = list()

idxMNFBL2 = list()

for(i in commonIndex){

  if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower2[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower2[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower2[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower3[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower3[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower3[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower4[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower4[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower4[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower5[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower5[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower5[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower6[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower6[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower6[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower7[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower7[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower7[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower8[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower8[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower8[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower9[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower9[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower9[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower10[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower10[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower10[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower11[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower11[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower11[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower12[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower12[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower12[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower13[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower13[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower13[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower14[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower14[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower14[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower15[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower15[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower15[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower16[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower16[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower16[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower17[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower17[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFBL2[i]=listAdj1$Apower17[edgelist[i,3],edgelist[i,2]]
  }

  if( (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower2[edgelist[i,3],edgelist[i,2]]) |
```

```r
listAdj1$Apower2[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower3[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower3[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower4[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower4[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower5[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower5[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower6[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower6[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower7[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower7[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower8[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower8[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower9[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower9[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower10[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower10[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower11[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower11[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower12[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower12[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower13[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower13[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower14[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower14[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower15[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower15[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower16[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower16[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj1$Apower17[edgelist[i,3],edgelist[i,2]]) |
listAdj1$Apower17[edgelist[i,3],edgelist[i,2]] == 0))
  ){idxMNFBL2[i] = i}

}
```

```r
MNFBL2 = unlist(MNFBL2)

idxMNFBL2 = unlist(idxMNFBL2)

length(idxMNFBL2)

## [1] 430
```

Now it’s time to compute the number of edges involved in multiple-edge subgraphs (table 2).

MFFL1

```r
MFFL1.pval.cor1 = 0
MFFL1.pval.cor.NA = 0
MFFL1.pval.cor2 = 0



for(i in idxMFFL1){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {MFFL1.pval.cor1=MFFL1.pval.cor1+1}
      if(all(logic2)) {MFFL1.pval.cor2=MFFL1.pval.cor2+1}
    }
  } else{
    if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
```

```
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){MFFL1.pval.cor1=MFFL1.pval.cor1+1}
      if(b){MFFL1.pval.cor2=MFFL1.pval.cor2+1}
    }
  }
}




for(i in idxMFFL1){
  logic = c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){

      logic[j]=any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
| SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
    }
    if(all(logic)){ MFFL1.pval.cor.NA = MFFL1.pval.cor.NA + 1 }

  }else{ if(any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))|
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05){
    index.pval.cor.NA = MFFL1.pval.cor.NA + 1}
  }
}

MFFL1.pval.cor1

## [1] 537

MFFL1.pval.cor.NA

## [1] 3061

MFFL1.pval.cor2

## [1] 270

length(idxMFFL1) - (MFFL1.pval.cor1 + MFFL1.pval.cor.NA + MFFL1.pval.cor2)

## [1] 4548
```

MNFBL2

```
MNFBL2.pval.cor1 = 0
MNFBL2.pval.cor.NA = 0
MNFBL2.pval.cor2 = 0
```

```r
for(i in idxMNFBL2){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {MNFBL2.pval.cor1=MNFBL2.pval.cor1+1}
      if(all(logic2)) {MNFBL2.pval.cor2=MNFBL2.pval.cor2+1}
    }
  } else{
    if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){MNFBL2.pval.cor1=MNFBL2.pval.cor1+1}
      if(b){MNFBL2.pval.cor2=MNFBL2.pval.cor2+1}
    }
  }
}



for(i in idxMNFBL2){
  logic = c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){

      logic[j]=any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
| SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
```

```
    }
    if(all(logic)){ MNFBL2.pval.cor.NA = MNFBL2.pval.cor.NA + 1 }

  }else{ if(any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))|
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05){
    index.pval.cor.NA = MNFBL2.pval.cor.NA + 1}
  }
}

MNFBL2.pval.cor1

## [1] 24

MNFBL2.pval.cor.NA

## [1] 181

MNFBL2.pval.cor2

## [1] 13

length(idxMNFBL2) - (MNFBL2.pval.cor1 + MNFBL2.pval.cor.NA +
MNFBL2.pval.cor2)

## [1] 212
```

MNFFL2

```
MNFFL2.pval.cor1 = 0
MNFFL2.pval.cor.NA = 0
MNFFL2.pval.cor2 = 0



for(i in idxMNFFL2){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {MNFFL2.pval.cor1=MNFFL2.pval.cor1+1}
      if(all(logic2)) {MNFFL2.pval.cor2=MNFFL2.pval.cor2+1}
    }
  } else{
    if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){MNFFL2.pval.cor1=MNFFL2.pval.cor1+1}
      if(b){MNFFL2.pval.cor2=MNFFL2.pval.cor2+1}
    }
  }
}




for(i in idxMNFFL2){
  logic = c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){

      logic[j]=any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
| SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
    }
    if(all(logic)){ MNFFL2.pval.cor.NA = MNFFL2.pval.cor.NA + 1 }

  }else{ if(any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))|
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05){
    index.pval.cor.NA = MNFFL2.pval.cor.NA + 1}
  }
}

MNFFL2.pval.cor1

## [1] 43

MNFFL2.pval.cor.NA

## [1] 254

MNFFL2.pval.cor2

## [1] 27
```

```r
length(idxMNFFL2) - (MNFFL2.pval.cor1 + MNFFL2.pval.cor.NA +
MNFFL2.pval.cor2)

## [1] 335
```

MPFBL1

```r
MPFBL1.pval.cor1 = 0
MPFBL1.pval.cor.NA = 0
MPFBL1.pval.cor2 = 0




for(i in idxMPFBL1){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {MPFBL1.pval.cor1=MPFBL1.pval.cor1+1}
      if(all(logic2)) {MPFBL1.pval.cor2=MPFBL1.pval.cor2+1}
    }
  } else{
    if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){MPFBL1.pval.cor1=MPFBL1.pval.cor1+1}
      if(b){MPFBL1.pval.cor2=MPFBL1.pval.cor2+1}
    }
  }
}
```

```
for(i in idxMPFBL1){
  logic = c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){

      logic[j]=any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
| SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
    }
    if(all(logic)){ MPFBL1.pval.cor.NA = MPFBL1.pval.cor.NA + 1 }

  }else{ if(any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))|
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05){
    index.pval.cor.NA = MPFBL1.pval.cor.NA + 1}
  }
}

MPFBL1.pval.cor1

## [1] 224

MPFBL1.pval.cor.NA

## [1] 916

MPFBL1.pval.cor2

## [1] 85

length(idxMPFBL1) - (MPFBL1.pval.cor1 + MPFBL1.pval.cor.NA +
MPFBL1.pval.cor2)

## [1] 1593
```

7.2.Computing the number of edges participating in MNFBL1, MPFBL2, MFFL2 and MNFFL1 multiple-edge subgraphs

The indices of edges which are involved in MNFBL1 subgraph are stored at **idxMNFBL1** object through the following code.

```
MNFBL1= list()
```

```
idxMNFBL1= list()

for(i in commonIndex){

  if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower2[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower2[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower2[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower3[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower3[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower3[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower4[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower4[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower4[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower5[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower5[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower5[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower6[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower6[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower6[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower7[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower7[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower7[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower8[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower8[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower8[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower9[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower9[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower9[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower10[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower10[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower10[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower11[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower11[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower11[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower12[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower12[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower12[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower13[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower13[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower13[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower14[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower14[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower14[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower15[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower15[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower15[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower16[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower16[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower16[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower17[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower17[edgelist[i,2],edgelist[i,3]] == 0)){
    MNFBL1[i]=listAdj2$Apower17[edgelist[i,2],edgelist[i,3]]
  }

  if( (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower2[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower2[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower3[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower3[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower4[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower4[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower5[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower5[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower6[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower6[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower7[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower7[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower8[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower8[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower9[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower9[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower10[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower10[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower11[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower11[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower12[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower12[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower13[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower13[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower14[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower14[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower15[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower15[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower16[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower16[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower17[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower17[edgelist[i,2],edgelist[i,3]] == 0))
  ){idxMNFBL1[i] = i}

}

MNFBL1 = unlist(MNFBL1)

idxMNFBL1 = unlist(idxMNFBL1)

length(idxMNFBL1)

## [1] 11532
```

The indices of edges involved in MPFBL2 subgraph are stored at **idxMPFBL2** object through the following code.

```
MPFBL2= list()

idxMPFBL2 = list()

for(i in commonIndex){

  if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower2[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower2[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower2[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower3[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower3[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower3[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower4[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower4[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower4[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower5[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower5[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower5[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower6[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower6[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower6[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower7[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower7[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower7[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower8[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower8[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower8[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower9[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower9[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower9[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower10[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower10[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower10[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower11[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower11[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower11[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower12[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower12[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower12[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower13[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower13[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower13[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower14[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower14[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower14[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower15[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower15[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower15[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower16[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower16[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower16[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower17[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower17[edgelist[i,3],edgelist[i,2]] == 0)){
    MPFBL2[i]=listAdj2$Apower17[edgelist[i,3],edgelist[i,2]]
  }

  if( (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower2[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower2[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower3[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower3[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower4[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower4[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower5[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower5[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower6[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower6[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower7[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower7[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower8[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower8[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower9[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower9[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower10[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower10[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower11[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower11[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower12[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower12[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower13[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower13[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower14[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower14[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower15[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower15[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower16[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower16[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower17[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower17[edgelist[i,3],edgelist[i,2]] == 0))
  ){idxMPFBL2[i] = i}

}


MPFBL2 = unlist(MPFBL2)

idxMPFBL2 = unlist(idxMPFBL2)

length(idxMPFBL2)

## [1] 973
```

The indices of edges involved in MFFL2 subgraph are stored at **idxMFFL2** object through the following code.

```
MFFL2 = list()

idxMFFL2 = list()

for(i in commonIndex){

  if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower2[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower2[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower2[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower3[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower3[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower3[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower4[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower4[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower4[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower5[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower5[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower5[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower6[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower6[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower6[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower7[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower7[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower7[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower8[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower8[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower8[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower9[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower9[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower9[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower10[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower10[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower10[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower11[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower11[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower11[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower12[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower12[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower12[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower13[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower13[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower13[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower14[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower14[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower14[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower15[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower15[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower15[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower16[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower16[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower16[edgelist[i,2],edgelist[i,3]]
  } else if(edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower17[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower17[edgelist[i,2],edgelist[i,3]] == 0)){
    MFFL2[i]=listAdj2$Apower17[edgelist[i,2],edgelist[i,3]]
  }

  if( (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower2[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower2[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower3[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower3[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower4[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower4[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower5[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower5[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower6[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower6[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower7[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower7[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower8[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower8[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower9[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower9[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower10[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower10[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower11[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower11[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower12[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower12[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower13[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower13[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower14[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower14[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower15[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower15[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower16[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower16[edgelist[i,2],edgelist[i,3]] == 0)) |
      (edgelist[i,4] == "-1" &
!(is.na(listAdj2$Apower17[edgelist[i,2],edgelist[i,3]]) |
listAdj2$Apower17[edgelist[i,2],edgelist[i,3]] == 0))
  ){idxMFFL2[i] = i}

}

MFFL2 = unlist(MFFL2)
```

```
idxMFFL2 = unlist(idxMFFL2)

length(idxMFFL2)

## [1] 2608
```

The indices of edges involved in MNFFL1 subgraph are stored at **MNFFL1** object through the following code.

```
MNFFL1 = list()

idxMNFFL1 = list()

for(i in commonIndex){

  if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower2[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower2[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower2[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower3[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower3[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower3[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower4[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower4[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower4[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower5[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower5[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower5[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower6[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower6[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower6[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower7[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower7[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower7[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower8[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower8[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower8[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower9[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower9[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower9[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower10[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower10[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower10[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower11[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower11[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower11[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower12[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower12[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower12[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower13[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower13[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower13[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower14[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower14[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower14[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower15[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower15[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower15[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower16[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower16[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower16[edgelist[i,3],edgelist[i,2]]
  } else if(edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower17[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower17[edgelist[i,3],edgelist[i,2]] == 0)){
    MNFFL1[i]=listAdj2$Apower17[edgelist[i,3],edgelist[i,2]]
  }

  if( (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower2[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower2[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower3[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower3[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower4[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower4[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower5[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower5[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower6[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower6[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower7[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower7[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower8[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower8[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower9[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower9[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower10[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower10[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower11[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower11[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower12[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower12[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower13[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower13[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower14[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower14[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower15[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower15[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower16[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower16[edgelist[i,3],edgelist[i,2]] == 0)) |
      (edgelist[i,4] == "1" &
!(is.na(listAdj2$Apower17[edgelist[i,3],edgelist[i,2]]) |
listAdj2$Apower17[edgelist[i,3],edgelist[i,2]] == 0))
  ){idxMNFFL1[i] = i}

}

MNFFL1 = unlist(MNFFL1)

idxMNFFL1 = unlist(idxMNFFL1)

length(idxMNFFL1)

## [1] 5267
```

Through the following code, the number of edges which are engaged in the multiple-edge subgraghs are computed.

MFFL2

```
MFFL2.pval.cor1 = 0
MFFL2.pval.cor.NA = 0
MFFL2.pval.cor2 = 0




for(i in idxMFFL2){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {MFFL2.pval.cor1=MFFL2.pval.cor1+1}
      if(all(logic2)) {MFFL2.pval.cor2=MFFL2.pval.cor2+1}
    }
  } else{
    if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){MFFL2.pval.cor1=MFFL2.pval.cor1+1}
      if(b){MFFL2.pval.cor2=MFFL2.pval.cor2+1}
    }
  }
}
```

```
for(i in idxMFFL2){
  logic = c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){

      logic[j]=any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
| SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
    }
    if(all(logic)){ MFFL2.pval.cor.NA = MFFL2.pval.cor.NA + 1 }

  }else{ if(any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))|
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05){
    index.pval.cor.NA = MFFL2.pval.cor.NA + 1}
  }
}

MFFL2.pval.cor1

## [1] 200

MFFL2.pval.cor.NA

## [1] 1057

MFFL2.pval.cor2

## [1] 106

length(idxMFFL2) - (MFFL2.pval.cor1 + MFFL2.pval.cor.NA + MFFL2.pval.cor2)

## [1] 1245
```

MNFBL1

```
MNFBL1.pval.cor1 = 0
MNFBL1.pval.cor.NA = 0
MNFBL1.pval.cor2 = 0



for(i in idxMNFBL1){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
```

```r
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] = 
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 & 
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] = 
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 & 
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {MNFBL1.pval.cor1=MNFBL1.pval.cor1+1}
      if(all(logic2)) {MNFBL1.pval.cor2=MNFBL1.pval.cor2+1}
    }
  } else{
    if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 & 
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 & 
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){MNFBL1.pval.cor1=MNFBL1.pval.cor1+1}
      if(b){MNFBL1.pval.cor2=MNFBL1.pval.cor2+1}
    }
  }
}




for(i in idxMNFBL1){
  logic = c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){

      logic[j]=any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
| SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
    }
    if(all(logic)){ MNFBL1.pval.cor.NA = MNFBL1.pval.cor.NA + 1 }

  }else{ if(any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))|
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05){
    index.pval.cor.NA = MNFBL1.pval.cor.NA + 1}
  }
}

MNFBL1.pval.cor1
```

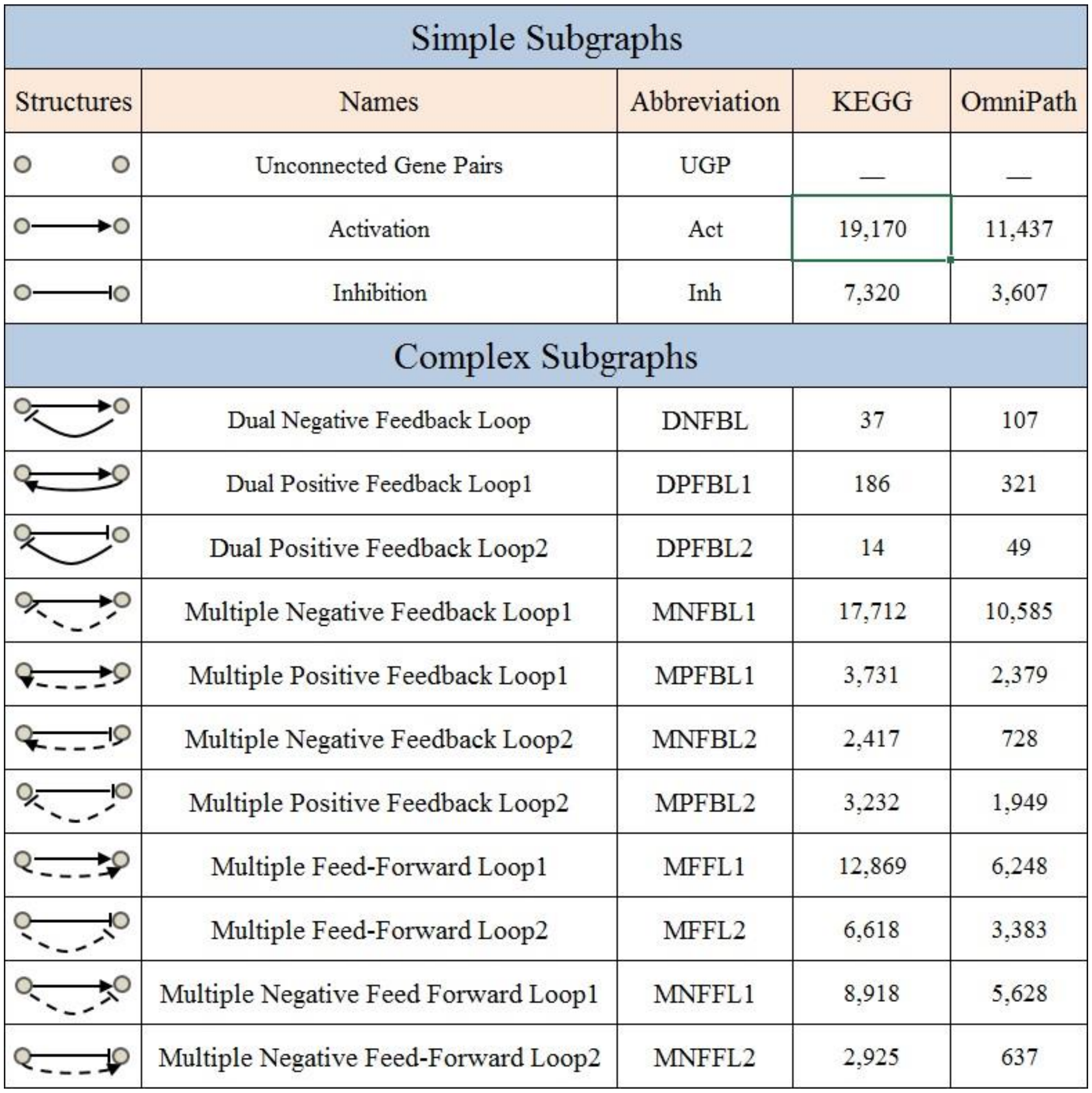
```
## [1] 773

MNFBL1.pval.cor.NA

## [1] 4205

MNFBL1.pval.cor2

## [1] 423

length(idxMNFBL1) - (MNFBL1.pval.cor1 + MNFBL1.pval.cor.NA +
MNFBL1.pval.cor2)

## [1] 6131
```

MNFFL1

```
MNFFL1.pval.cor1 = 0
MNFFL1.pval.cor.NA = 0
MNFFL1.pval.cor2 = 0



for(i in idxMNFFL1){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {MNFFL1.pval.cor1=MNFFL1.pval.cor1+1}
      if(all(logic2)) {MNFFL1.pval.cor2=MNFFL1.pval.cor2+1}
    }
  } else{
    if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){MNFFL1.pval.cor1=MNFFL1.pval.cor1+1}
      if(b){MNFFL1.pval.cor2=MNFFL1.pval.cor2+1}
    }
  }
}




for(i in idxMNFFL1){
  logic = c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){

      logic[j]=any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
| SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
    }
    if(all(logic)){ MNFFL1.pval.cor.NA = MNFFL1.pval.cor.NA + 1 }

  }else{ if(any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))|
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05){
    index.pval.cor.NA = MNFFL1.pval.cor.NA + 1}
  }
}

MNFFL1.pval.cor1

## [1] 393

MNFFL1.pval.cor.NA

## [1] 1820

MNFFL1.pval.cor2

## [1] 193

length(idxMNFFL1) - (MNFFL1.pval.cor1 + MNFFL1.pval.cor.NA +
MNFFL1.pval.cor2)

## [1] 2861
```

MPFBL2

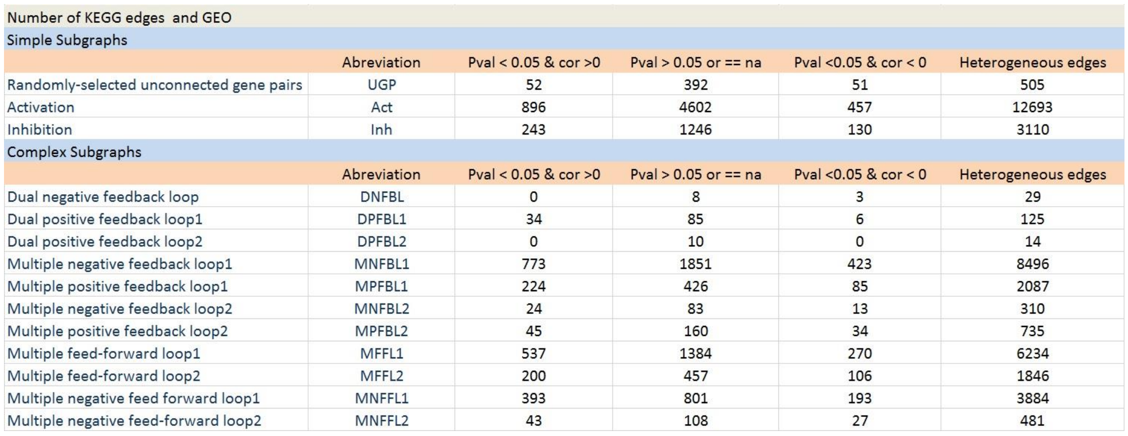
```
MPFBL2.pval.cor1 = 0
MPFBL2.pval.cor.NA = 0
MPFBL2.pval.cor2 = 0



for(i in idxMPFBL2){
  logic =c()
  if(SignalingNet[[i]]$length > 1){

    for(j in 1:SignalingNet[[i]]$length){
      logic[j]=
!any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
    }
    if(all(logic)){
      logic1 = c()
      logic2 = c()
      for(j in 1:SignalingNet[[i]]$length){

        logic1[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor > 0
        logic2[j] =
SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis[[j]]$pearson$pearsoncor < 0
      }
      if(all(logic1)) {MPFBL2.pval.cor1=MPFBL2.pval.cor1+1}
      if(all(logic2)) {MPFBL2.pval.cor2=MPFBL2.pval.cor2+1}
    }
  } else{
    if(!any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))){
      a = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor > 0
      b = SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval < 0.05 &
SignalingNet[[i]]$corAnalysis$pearson$pearsoncor < 0
      if(a){MPFBL2.pval.cor1=MPFBL2.pval.cor1+1}
      if(b){MPFBL2.pval.cor2=MPFBL2.pval.cor2+1}
    }
  }
}




for(i in idxMPFBL2){
  logic = c()
  if(SignalingNet[[i]]$length > 1){
    for(j in 1:SignalingNet[[i]]$length){

      logic[j]=any(unlist(lapply(SignalingNet[[i]]$corAnalysis[[j]],is.na)))
| SignalingNet[[i]]$corAnalysis[[j]]$pearson$adjusted.pearsonpval > 0.05
    }
    if(all(logic)){ MPFBL2.pval.cor.NA = MPFBL2.pval.cor.NA + 1 }

  }else{ if(any(unlist(lapply(SignalingNet[[i]]$corAnalysis,is.na)))|
SignalingNet[[i]]$corAnalysis$pearson$adjusted.pearsonpval > 0.05){
    index.pval.cor.NA = MPFBL2.pval.cor.NA + 1}
  }
}

MPFBL2.pval.cor1

## [1] 45

MPFBL2.pval.cor.NA

## [1] 355

MPFBL2.pval.cor2

## [1] 34

length(idxMPFBL2) - (MPFBL2.pval.cor1 + MPFBL2.pval.cor.NA +
MPFBL2.pval.cor2)

## [1] 539
```



Simple Subgraphs				
Structures	Names	Abbreviation	KEGG	OmniPath
	Unconnected Gene Pairs	UGP	—	—
	Activation	Act	19,170	11,437
	Inhibition	Inh	7,320	3,607
Complex Subgraphs				
	Dual Negative Feedback Loop	DNFBL	37	107
	Dual Positive Feedback Loop1	DPFBL1	186	321
	Dual Positive Feedback Loop2	DPFBL2	14	49
	Multiple Negative Feedback Loop1	MNFBL1	17,712	10,585
	Multiple Positive Feedback Loop1	MPFBL1	3,731	2,379
	Multiple Negative Feedback Loop2	MNFBL2	2,417	728
	Multiple Positive Feedback Loop2	MPFBL2	3,232	1,949
	Multiple Feed-Forward Loop1	MFFL1	12,869	6,248
	Multiple Feed-Forward Loop2	MFFL2	6,618	3,383
	Multiple Negative Feed Forward Loop1	MNFFL1	8,918	5,628
	Multiple Negative Feed-Forward Loop2	MNFFL2	2,925	637

Table1

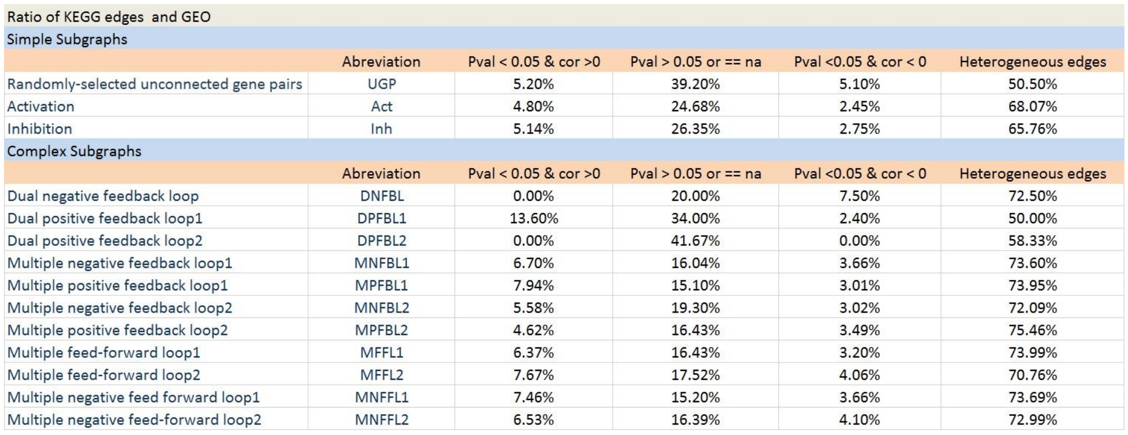
Results



Number of KEGG edges and GEO

Simple Subgraphs					
	Abreviation	Pval < 0.05 & cor >0	Pval > 0.05 or == na	Pval <0.05 & cor < 0	Heterogeneous edges
Randomly-selected unconnected gene pairs	UGP	52	392	51	505
Activation	Act	896	4602	457	12693
Inhibition	Inh	243	1246	130	3110
Complex Subgraphs					
	Abreviation	Pval < 0.05 & cor >0	Pval > 0.05 or == na	Pval <0.05 & cor < 0	Heterogeneous edges
Dual negative feedback loop	DNFBL	0	8	3	29
Dual positive feedback loop1	DPFBL1	34	85	6	125
Dual positive feedback loop2	DPFBL2	0	10	0	14
Multiple negative feedback loop1	MNFBL1	773	1851	423	8496
Multiple positive feedback loop1	MPFBL1	224	426	85	2087
Multiple negative feedback loop2	MNFBL2	24	83	13	310
Multiple positive feedback loop2	MPFBL2	45	160	34	735
Multiple feed-forward loop1	MFFL1	537	1384	270	6234
Multiple feed-forward loop2	MFFL2	200	457	106	1846
Multiple negative feed forward loop1	MNFFL1	393	801	193	3884
Multiple negative feed-forward loop2	MNFFL2	43	108	27	481

Table2



Ratio of KEGG edges and GEO

Simple Subgraphs					
	Abreviation	Pval < 0.05 & cor >0	Pval > 0.05 or == na	Pval <0.05 & cor < 0	Heterogeneous edges
Randomly-selected unconnected gene pairs	UGP	5.20%	39.20%	5.10%	50.50%
Activation	Act	4.80%	24.68%	2.45%	68.07%
Inhibition	Inh	5.14%	26.35%	2.75%	65.76%
Complex Subgraphs					
	Abreviation	Pval < 0.05 & cor >0	Pval > 0.05 or == na	Pval <0.05 & cor < 0	Heterogeneous edges
Dual negative feedback loop	DNFBL	0.00%	20.00%	7.50%	72.50%
Dual positive feedback loop1	DPFBL1	13.60%	34.00%	2.40%	50.00%
Dual positive feedback loop2	DPFBL2	0.00%	41.67%	0.00%	58.33%
Multiple negative feedback loop1	MNFBL1	6.70%	16.04%	3.66%	73.60%
Multiple positive feedback loop1	MPFBL1	7.94%	15.10%	3.01%	73.95%
Multiple negative feedback loop2	MNFBL2	5.58%	19.30%	3.02%	72.09%
Multiple positive feedback loop2	MPFBL2	4.62%	16.43%	3.49%	75.46%
Multiple feed-forward loop1	MFFL1	6.37%	16.43%	3.20%	73.99%
Multiple feed-forward loop2	MFFL2	7.67%	17.52%	4.06%	70.76%
Multiple negative feed forward loop1	MNFFL1	7.46%	15.20%	3.66%	73.69%
Multiple negative feed-forward loop2	MNFFL2	6.53%	16.39%	4.10%	72.99%

Table3