

```

1 '''
2 Summary: This notebook abandons the dense WGCNA approach to construct a sparse, topological gene regulatory network
3 using the igraph package. By applying a dynamic signed threshold to a Spearman correlation matrix and executing
4 Prim's algorithm on the raw weights, it forces the construction of a Minimum Spanning Tree (MST). It then isolates
5 topological bottlenecks by stripping edge weights during community detection, calculates formal mathematical centrality
6 metrics (degree, closeness, betweenness, and eigenvector), and subsets the highest-ranking "bottleneck" genes from each
7 community for targeted GO and KEGG enrichment.
8 '''

```

```

1 %load_ext rpy2.ipynthon
2 from google.colab import drive
3 drive.mount('/content/drive')

```

```

1 %%R -o Idx_new -o DER_01 -o DER_03a -o DER_03b -o DER_tmp -o capstone
2
3 install.packages("R.matlab")
4 library(R.matlab)
5 library(dplyr)
6
7 DER_tmp <- read.delim("/content/drive/My Drive/datasets/DER-02_PEC_Gene_expression_matrix_TPM.txt")
8 capstone_tmp <- read.csv("/content/drive/My Drive/datasets/PEC_capstone_data_map_clinical.csv")
9 capstone <- capstone_tmp[, c(1, 2, 6, 7, 8)]
10
11 DER_01 <- read.delim("/content/drive/My Drive/datasets/DER-02_PEC_Gene_expression_matrix_TPM.txt")
12 DER_01[,1] <- gsub("c(", "", gsub("\\.\\d+", "", DER_01[,1]), fixed="TRUE")
13
14 scz_data1 <- readMat("/content/drive/My Drive/datasets/scz_data1.mat")
15
16 scz_data1f <- data.frame("X.geneIds"=matrix(unlist(scz_data1$X.geneIds), nrow=length(scz_data1$X.geneIds), byrow=TRUE), strin
17 scz_data1f$X.geneIds <- as.character(scz_data1f$X.geneIds)
18 scz_data1f$X.geneIds <- sub("[0-9]*", "", scz_data1f$X.geneIds)
19 genes <- scz_data1f$X.geneIds
20 crs3 <- list()
21 crs3 <- DER_01[DER_01[,1] %in% genes,1]
22 Idx_new <- which(DER_01[,1] %in% genes)
23
24 install.packages("tidyverse")
25 library(tidyverse)
26
27 DER_03a <- DER_01[,1] #__ gene names
28 DER_03b <- colnames(DER_01)[-1] #__ sample names
29 DER_01 <- t(scale(t(DER_01[, -1]))) #__ scaling the data (otherwise all of the highly expressed genes
30 #will cluster together even if they have different patterns
31 #among the samples)
32 library(data.table)
33 BrainSeq <- fread('https://raw.githubusercontent.com/LieberInstitute/brainseq_phase2/master/BrainSeq_Phase2_phenotype_data_s
34 head(BrainSeq)
35

```

```

1 import numpy
2 import numpy as np
3 import pandas as pd
4
5 %R DEX_Genes <- read.csv('/content/drive/My Drive/datasets/DER-13_Disorder_DEX_Genes.csv')
6 %R dim(DEX_Genes)
7 %R DEX_SCZ <- DEX_Genes[grep("SCZ", DEX_Genes$Disorder.DGE_RegulationDirection), ]
8 %R dim(DEX_SCZ)
9 %R -o DEX_SCZ
10 arr3 = np.array([])
11 arr3 = DEX_SCZ.values[:,1]
12 seen = set() #_____ remove duplicates
13 uniq = []
14 for x in arr3:
15     if x not in seen:
16         uniq.append(x)
17         seen.add(x) #_____ (4821,)
18 if (len(DEX_SCZ) == len(uniq)):
19     print("There are no duplicates")
20
21 arr4 = np.array([])
22 arr4 = DER_03a
23 xz2, z1_ind, z2_ind = np.intersect1d(arr3, arr4,

```

24
25

return_indices=True)

```
1 import numpy #_____ Divide tissues and subjects to brain/peripheral
2 import numpy as np # and CTL/SCZ, respectively.
3 import pandas as pd
4 from sklearn.preprocessing import MinMaxScaler
5 from sklearn.feature_selection import VarianceThreshold
6 from sklearn.preprocessing import StandardScaler
7 from scipy.stats import gaussian_kde
8
9 %R library(data.table)
10 %R BrainSeq <- fread('https://raw.githubusercontent.com/LieberInstitute/brainseq_phase2/master/BrainSeq_Phase2_phenotype_c
11 %R head(BrainSeq)
12 %R E <- readRDS('/content/drive/My Drive/datasets/expression.rds')
13 %R gdt <- readRDS('/content/drive/My Drive/datasets/gdt.rds')
14 %R samp <- fread('https://storage.googleapis.com/gtex_analysis_v7/annotations/GTex_v7_Annotations_SampleAttributesDS.txt')
15 %R subj <- fread('https://storage.googleapis.com/gtex_analysis_v7/annotations/GTex_v7_Annotations_SubjectPhenotypesDS.txt')
16 %R samp[, SUBJID := gsub('^-*)-([^-]*)-.*', '\\1-\\2', SAMPID)]
17 %R sdt <- merge(samp, subj, by='SUBJID')
18 %R sdt <- merge(samp, subj, by='SUBJID')[SAMPID %in% rownames(E)]
19 #_____
20
21 %R -o BrainSeq
22 %R -o DER_01
23
24 #1.____BrainSeq[~BrainSeq.Region.str.contains("HIPPO")] #_____ Check for bias due to Tissue/Sex/Race.
25 #2.____BrainSeq[~BrainSeq.Race.str.contains("AA")] #_____ DLPFC/HIPPO # Replace each line to the line below "3.____
26 #3.____BrainSeq[~BrainSeq.Sex.str.contains("F")] #_____ CAUC/AA #_____ M/F
27 BrSeq = pd.DataFrame(BrainSeq)
28 #capstone = pd.DataFrame(capstone)
29
30 DER01 = pd.DataFrame(DER_01)
31 DER01.columns = DER_03b
32 DER01 = np.transpose(DER01)
33 DER01.columns = DER_03a
34 DER01 = np.transpose(DER01)
35
36 #_____ I. BrainSeq DATASET(from Lieber Institute)
37
38 arr20 = np.array([])
39 arr20 = BrSeq['BrNum']
40 arr21 = np.array([])
41 arr21 = DER01.columns
42 xy21, x21_ind, y21_ind = np.intersect1d(arr20, arr21,
43 return_indices=True)
44 data1 = []
45 for i in x21_ind:
46 data1.append([BrSeq['BrNum'][i], BrSeq['Region'][i], BrSeq['Dx'][i],
47 BrSeq['Sex'][i], BrSeq['Race'][i], BrSeq['RIN'][i], BrSeq['Age'][i]])
48 DER01_BrSeq = pd.DataFrame(data1, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G'])
49 #DER01_BrSeq = DER01_BrSeq[~DER01_BrSeq.B.str.contains("HIPPO")]
50 #_____
51 %R -o sdt
52 GTx = pd.DataFrame(sdt)
53
54 arr22 = np.array([])
55 arr22 = GTx['SUBJID']
56 xy22, x22_ind, y22_ind = np.intersect1d(arr22, arr21,
57 return_indices=True)
58 data2 = []
59 for i in x22_ind:
60 data2.append([GTx['SUBJID'][i], GTx['SMTSD'][i]])
61 df_GTx = pd.DataFrame(data2, columns=['A', 'B'])
62 #_____
63 #FER01 = DER01.columns.to_series().str.contains('CMC')
64 #y23_ind = np.array([i for i, x in enumerate(FER01) if x])
65 #FER02 = DER01.columns.to_series().str.contains('Br')
66 #y24_ind = np.array([i for i, x in enumerate(FER02) if x])
67 #gtx_br = [*y23_ind, *y24_ind]
68
69 #gtx_br = [*y21_ind, *y23_ind, *y22_ind]
70 gtx_br = [*y21_ind, *y22_ind]
71
72 n13_train_scz = pd.DataFrame(DER01)
73 n14_train_scz = n13_train_scz.values[list(z2_ind),] #_____ Filter rows by DEGs or (igraph)...(1/2)
```

```

74 n15_train_scz = n14_train_scz[:,gtx_br] #_____ Filter samples
75
76 #_____
77
78 #n13_train_scz_min = n15_train_scz.astype(float) #_____1. No transformation
79
80 ###medianValue = median(logData); #_____2. Median centering data before log2-trans
81 ###medianCtrData = logData-medianValue
82
83 n13_train_scz_min = np.log(n15_train_scz.astype(float)+1) #_____3. Log2 transformation
84
85 #n13_train_scz_min <- zFPKM(n15_train_scz.astype(float)) #_____4. Z-scale transformation (in R, from zFPK
86
87 n13_train_scz_min = np.transpose(n13_train_scz_min) #_____ Shape the array as (samples, genes)
88
89 n13_train_scz_min[np.isnan(n13_train_scz_min)] = 0 #(57820, 414) #_____ Replace NAs with zeroes
90 pd.DataFrame(n13_train_scz_min).isnull().sum()
91
92 print(n13_train_scz_min.shape)
93 jdx = np.argwhere(np.all(n13_train_scz_min[... ,:] == 0, axis=0)) #_____ Remove zero columns
94 n13_train_scz_min = np.delete(n13_train_scz_min, jdx, axis=1)
95 print(n13_train_scz_min.shape)
96
97
98
99 #n13_train_scz_min = pd.DataFrame(n13_train_scz_min)
100 #transform = VarianceThreshold(0) #_____ Remove genes with variance
101 print(n13_train_scz_min.shape) # over a threshold
102 DER_04a = np.array(DER_03a)[list(z2_ind)] #_____ Filter rows by DEGs or (igraph)...(2/2)
103 DER_05a = np.delete(DER_04a, jdx)
104 #DER_06a = DER_05a[VarianceThreshold().fit(n13_train_scz_min).get_support()] #___ Final Gene headers!!!!
105 #n13_train_scz_min = n13_train_scz_min.iloc[:,VarianceThreshold().fit(n13_train_scz_min).get_support()]
106
107 f = VarianceThreshold().fit(n13_train_scz_min)
108 DER_06a = DER_05a[f.variances_ > 0.0] #___ Final Gene headers!!!!
109 print(DER_06a.shape)
110 n13_train_scz_min = n13_train_scz_min[:, f.variances_ > 0.0]
111 print(n13_train_scz_min.shape)
112
113 %R -i n13_train_scz_min #_____ Remove zero columns
114
115 #_____ Traits (CTL/SCZ)
116
117 data4 = [] #I. BrainSeq DATASET(from Lieber Institute)
118 for i in DER01_BrSeq['B']:
119     if i == 'DLPFC':
120         k = 0
121     if i == 'HIPPO':
122         k = 1
123     data4.append(k)
124 DER01_BrSeq6 = pd.DataFrame(data4, columns=['region'])
125
126 data4 = []
127 for i in DER01_BrSeq['C']:
128     if i == 'Control':
129         k = 0
130     if i == 'Schizo':
131         k = 1
132     data4.append(k)
133 DER01_BrSeq1 = pd.DataFrame(data4, columns=['diag'])
134
135 data4 = []
136 for i in DER01_BrSeq['D']:
137     if i == 'F':
138         k = 0
139     if i == 'M':
140         k = 1
141     data4.append(k)
142 DER01_BrSeq2 = pd.DataFrame(data4, columns=['sex'])
143
144 data4 = []
145 for i in DER01_BrSeq['E']:
146     if i == 'CAUC':
147         k = 1
148     else:
149         k = 0
150     data4.append(k)

```

```

151 DER01_BrSeq3 = pd.DataFrame(data4, columns=['ethn'])
152
153 data4 = []
154 for i in DER01_BrSeq['F']:
155     data4.append(i)
156 DER01_BrSeq4 = pd.DataFrame(data4, columns=['rin'])
157
158 data4 = []
159 for i in DER01_BrSeq['G']:
160     data4.append(i)
161 DER01_BrSeq5 = pd.DataFrame(data4, columns=['age'])
162
163 DER01_BrSeq_Tot = pd.concat([DER01_BrSeq1, DER01_BrSeq2, DER01_BrSeq3,
164                             DER01_BrSeq4, DER01_BrSeq5, DER01_BrSeq6], axis=1)
165
166 n13_trait_path1 = []
167 n13_trait_path1 = pd.DataFrame(DER01_BrSeq_Tot)
168
169 n13_trait_path = n13_trait_path1
170 %R -i n13_trait_path
171
172 del DER01_BrSeq #_____ Clear unused variables
173 #del DER01_BrFin
174 del DER01
175 del n13_train_scz
176 del n14_train_scz
177 del n15_train_scz
178

```

```

1 # _____ Defining number of CTLs and SCZs
2 # before removing outliers!!!!!!!!!!
3 # Use it as input to the PCA below
4
5
6 #I. BrainSeq DATASET(from Lieber Institute)
7 DER01_BrSeq = pd.DataFrame(data1, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G'])
8 #DER01_BrSeq = DER01_BrSeq[~DER01_BrSeq.B.str.contains("HIPPO")]
9
10 FER05 = DER01_BrSeq['C'].str.contains('Con')
11 z23_ind = np.array([i for i, x in enumerate(FER05) if x])
12
13 FER06 = DER01_BrSeq['C'].str.contains('Sc')
14 z24_ind = np.array([i for i, x in enumerate(FER06) if x])
15
16 gtx_all = [*z23_ind, *z24_ind]
17 #print(DER01_BrSeq.values[gtx_all])
18
19 n13_pca_scz_min = n13_train_scz_min[gtx_all,]
20 n13_pca_path = n13_trait_path.iloc[gtx_all]
21 #n13_pca_path = n13_trait_path['D'][gtx_all]
22 %R -i n13_pca_path
23
24 DER01_BrSeq['C'].value_counts() #_____ CTLs and SCZs
25

```

```

1 %%R -i n13_pca_scz_min
2 # _____ plot PCA before removing outliers!!!!!!!!!!
3
4 library(ggplot2)
5
6 n13_pca_scz_min <- n13_pca_scz_min[,which(apply(n13_pca_scz_min, 2, var) != 0)]#_Remove zero variance columns from the new
7 genes=paste("gene",seq(1:dim(n13_pca_scz_min)[2]), sep="")
8 colnames(n13_pca_scz_min)=genes
9 row.names(n13_pca_scz_min)=c(paste0("ctl_",seq(1:261)), paste0("scz",seq(1:153))) #__Python command: DER01_BrSeq['C'].val
10 condition1=rep(c("ctl"), each=261)
11 condition2=rep(c("scz"), each=153)
12 condition1 <- append(condition1, condition2)
13 length(condition1)
14
15 #pca_data=prcomp(n13_train_scz_min, center = TRUE, scale = TRUE)
16 #pca_data_perc=round(100*pca_data$sdev^2/sum(pca_data$sdev^2),1)
17 #df_pca_data=data.frame(PC1 = pca_data$x[,1], PC2 = pca_data$x[,2], sample = row.names(n13_train_scz_min), condition1=condi
18 #ggplot(df_pca_data, aes(PC1,PC2, color = condition1))+
19 #     geom_point(size=8)+
20 #     labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))
21

```

```

22                                                                 #____ PCA for Z-scaled data
23 pca_data=prcomp(n13_pca_scz_min, center = TRUE, scale = TRUE)
24 pca_data_perc=round(100*pca_data$sdev^2/sum(pca_data$sdev^2),1)
25 df_pca_data=data.frame(PC1 = pca_data$x[,1], PC2 = pca_data$x[,2], sample = row.names(n13_pca_scz_min), condition1=condition1)
26
27                                                                 #_____ Plot samples
28 ggplot(df_pca_data, aes(PC1,PC2, color = condition1))+
29   geom_point(size=2)+
30   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))
31

```

```

1 %%R -i n13_pca_path
2
3 datExpr = n13_pca_scz_min
4
5 nGenes <- ncol(datExpr)
6 nSamples <- nrow(datExpr)
7
8 datTraits <- n13_pca_path
9
10 rm(list=ls()[! ls() %in% c("datExpr", "nGenes", "datTraits", "nSamples")])
11 ls()
12

```

```

1 %%R
2
3 # Setting the colors to match the regional colors
4 region_labels <- RColorBrewer::brewer.pal(6, "Set1")[1:3]
5
6 # Making sure the regional colors match what has been used throughout and making a column with the hex codes.
7
8 datTraits <- datTraits %>%
9   mutate(cell.names = row.names(datTraits)) %>%
10   mutate(x = ifelse(datTraits$sex == "1", region_labels[1],
11                     ifelse(datTraits$sex == "0", region_labels[2], NA)))
12
13 datTraits <- datTraits %>%
14   mutate(cell.names = row.names(datTraits)) %>%
15   mutate(x = ifelse(datTraits$ethn == "1", region_labels[1],
16                     ifelse(datTraits$ethn == "0", region_labels[2], NA)))
17
18 datTraits <- datTraits %>%
19   mutate(cell.names = row.names(datTraits)) %>%
20   mutate(x = ifelse(datTraits$region == "1", region_labels[1],
21                     ifelse(datTraits$region == "0", region_labels[2], NA)))
22
23 datTraits <- datTraits %>%
24   mutate(cell.names = row.names(datTraits)) %>%
25   mutate(x = ifelse(datTraits$diag == "1", region_labels[1],
26                     ifelse(datTraits$diag == "0", region_labels[2], NA)))
27
28 xs=quantile(datTraits$age,c(0,1/3,2/3,1))
29 xs[1]=xs[1]-.00005
30 datTraits <- datTraits %>% mutate(category=cut(age, breaks=xs,
31 labels=c("low","middle","high"),ordered_result = TRUE))
32 boxplot(datTraits$age~datTraits$category,col=3:5)
33
34 datExpr <- datExpr[,which(apply(datExpr, 2, var) != 0)]
35 datExpr = t(datExpr)
36 colnames(datExpr) <- c(row.names(datTraits))
37
38 datTraits$diag <- as.factor(datTraits$diag)
39 datTraits$ethn <- as.factor(datTraits$ethn)
40 datTraits$sex <- as.factor(datTraits$sex)
41 datTraits$age <- as.factor(datTraits$category)
42 datTraits$region <- as.factor(datTraits$region)
43
44 datExpr = t(datExpr)

```

```

1 %%R                                                                 #_____ SVA analysis
2
3 if (!requireNamespace('BiocManager', quietly = TRUE))
4   install.packages('BiocManager')
5
6 BiocManager::install('sva')

```

```

7 library(sva)
8
9 mod0 = model.matrix(~as.factor(ethn)+as.factor(sex)+as.factor(age)+as.factor(region), data=datTraits)
10 mod = model.matrix(~as.factor(diag)+as.factor(ethn)+as.factor(sex)+as.factor(age)+as.factor(region), data=datTraits)
11 n.sv = num.sv(t(datExpr),mod,method="leek")
12 svobj = sva(t(datExpr),mod,mod0,n.sv=n.sv)
13
14 cleanY = function(y, mod, svobj) {
15   X = cbind(mod, svobj)
16   Hat = solve(t(X) %*% X) %*% t(X)
17   beta = (Hat %*% t(y))
18   rm(Hat)
19   gc()
20   P = ncol(mod)
21   return(y - t(as.matrix(X[,-c(1:P)]) %*% beta[-c(1:P),]))
22 }
23
24 datExpr_cln <- cleanY(t(datExpr), mod, svobj$sv)
25 datExpr_cln <- t(datExpr_cln)
26 datExpr = datExpr_cln
27 nGenes <- ncol(datExpr)
28 nSamples <- nrow(datExpr)
29

```

```

1 ##R
2
3 rm(list=ls()[! ls() %in% c("datExpr", "nGenes", "datTraits", "nSamples")])
4 ls()
5

```

```

1 ##R
2 #_____ plot PCA after removing outliers!!!!!!!!!!
3
4 library(ggplot2)
5
6 row.names(datExpr)=c(paste0("ctl_",seq(1:261)), paste0("scz",seq(1:153))) #___Python command: DER01_BrSeq['C'].value_count
7 condition1=rep(c("ctl"), each=261)
8 condition2=rep(c("scz"), each=153)
9 condition1 <- append(condition1, condition2)
10 length(condition1)
11
12 #_____ PCA for Z-scaled data
12 pca_data=prcomp(datExpr, center = TRUE, scale = TRUE)
13 pca_data_perc=round(100*pca_data$sdev^2/sum(pca_data$sdev^2),1)
14 df_pca_data=data.frame(PC1 = pca_data$x[,1], PC2 = pca_data$x[,2], sample = row.names(datExpr), condition1=condition1)
15
16 #_____Plot samples
17 ggplot(df_pca_data, aes(PC1,PC2, color = condition1))+
18   geom_point(size=2)+
19   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))
20

```

```

1 ##R -i DER_06a
2
3 colnames(datExpr) <- DER_06a
4
5 #Corr_ctl2 <- as.matrix(as.dist(cor(datExpr[1:261,], method="spearman"))) #_____ Correlations for CTL network
6 Corr_ctl2 <- as.matrix(as.dist(cor(datExpr[262:414,], method="spearman"))) #_____ Correlations for SCZ network
7 #_____method="pearson", "kendall", "spearman"
8
9 #save(Corr_ctl2, file="/content/drive/My Drive/datasets/Corr_ctl_net.Rdata")
10 #save(Corr_ctl2, file="/content/drive/My Drive/datasets/Corr_scz_net.Rdata")
11
12 #_____
13 #load("/content/drive/My Drive/datasets/Corr_ctl_net.Rdata")
14 #load("/content/drive/My Drive/datasets/Corr_scz_net.Rdata")
15 Corr_ctl2 <- ((Corr_ctl2-min(Corr_ctl2))/(max(Corr_ctl2)-min(Corr_ctl2)))*2 - 1
16
17 set.seed(1234)
18
19 install.packages("igraph")
20 library(igraph)
21 #_____
22 ctl1 <- graph.adjacency(
23   Corr_ctl2,
24   mode="undirected",

```

```

25 weighted=TRUE,
26 diag=FALSE)
27 ct11 <- delete_edges(ct11, E(ct11)[which(E(ct11)$weight < max(E(ct11)$weight)*0.8 & E(ct11)$weight > min(E(ct11)$weight)*0.
28 ct11 <- igraph::simplify(ct11, remove.multiple=TRUE, remove.loops=FALSE)
29 #save(ct11, file="/content/drive/My Drive/datasets/ctl_net.Rdata")
30
31 E(ct11)[which(E(ct11)$weight<0)]$color <- "darkblue"
32 E(ct11)[which(E(ct11)$weight>0)]$color <- "darkred"
33
34 G2 <- igraph::delete.vertices(ct11, igraph::graph.strength(ct11)==0)
35
36 ct11 <- induced_subgraph(
37   G2, V(G2)[components(G2)$membership == which.max(components(G2)$csize)]
38 )
39 print(is.connected(ct11))
40
41 V(ct11)$shape <- "sphere"
42 V(ct11)$color <- "skyblue"
43 V(ct11)$vertex.frame.color <- "white"
44
45 M2.subgraph <- mst(ct11, algorithm="prim")
46 print(is.connected(M2.subgraph))
47
48 E(M2.subgraph)$weight <- abs(E(M2.subgraph)$weight)
49

```

```

1 %%R                                     #_____ RUN FIRST
2 '''
3 M2.subgraph <- induced_subgraph(
4   G2, V(G2)[components(G2)$membership == which.max(components(G2)$csize)]
5 )
6 print(is.connected(M2.subgraph))
7
8 V(M2.subgraph)$shape <- "sphere"
9 V(M2.subgraph)$color <- "skyblue"
10 V(M2.subgraph)$vertex.frame.color <- "white"
11
12 E(M2.subgraph)$weight <- abs(E(M2.subgraph)$weight)
13 '''

```

```

1 %%R -i n13_pca_scz_min                                     #_____ RUN SECOND
2 '''
3 #_____ I. Normality tests
4
5 #x = datExpr[1:261,]; qqnorm(x); qqline(x,col='red')          #_____ CTL network
6 #hist(x)
7 #boxplot(x)
8
9
10 x = datExpr[262:414,]; qqnorm(x); qqline(x,col='red')      #_____ SCZ network
11 hist(x)
12 boxplot(x)
13
14 #_____ II. CTL and SCZ Networks Various Distribut
15
16 hist(igraph::degree(M2.subgraph), col="lightblue", breaks=10,#xlim=c(0,400),
17     xlab="Vertex Degree", ylab="Frequency", main="Histogram of node degree")
18
19 hist(igraph::closeness(M2.subgraph), col="lightblue", breaks=10,#xlim=c(0,400),
20     xlab="Closeness", ylab="Frequency", main="Histogram of node closeness")
21
22 hist(igraph::betweenness(M2.subgraph), col="lightblue", breaks=10,#xlim=c(0,400),
23     xlab="Betweenness", ylab="Frequency", main="Histogram of node betweenness")
24
25 hist(igraph::evcent(M2.subgraph)$vector, col="lightblue", breaks=10,#xlim=c(0,400),
26     xlab="EigenVector", ylab="Frequency", main="Histogram of node eigenVector")
27 '''

```

```

1 %%R
2
3 install.packages("GGally")
4 install.packages("ggplot2")
5 install.packages("ggdendro")
6 library(GGally)
7 library(ggplot2)

```

```

8 library(ggdendro)
9
10 #M2_LGL = layout_nicely(M2.subgraph)
11 #M2_LGL = layout_fruchterman_reingold(M2.subgraph)
12 M2_LGL = layout_with_lgl(M2.subgraph)
13 plot(M2.subgraph, vertex.size = 2, vertex.frame.color = NULL, layout=M2_LGL, asp=FALSE,
14 vertex.label=NA, edge.curved=TRUE, edge.arrow.size=0.1, vertex.label.dist = 0.5, edge.width = 0.5)
15
16 #
17 #M2.subgraph.communities <- cluster_louvain(as.undirected(M2.subgraph), weights = NULL) #weights = E(M2.subgraph)$w
18 #louvain_sizesComm <- sizes(M2.subgraph.communities)
19 #louvain_numComm <- length(louvain_sizesComm)
20 #louvain_modularity <- modularity(M2.subgraph.communities)
21 #print(louvain_numComm)
22 ##print(louvain_sizesComm)
23 #print(louvain_modularity)
24
25 M2.subgraph.communities <- edge.betweenness.community(as.undirected(M2.subgraph),
26 weights = NULL, modularity = TRUE, membership = TRUE) ##>% cut_at(no = 23)
27
28 #M2.subgraph.communities$degree <- (igraph::degree(M2.subgraph)[M2.subgraph.communities$names])
29 #M2.subgraph.communities$cluster <- unname(ave(M2.subgraph.communities$degree,
30 # M2.subgraph.communities$membership,
31 # FUN=function(x)names(x)[which.max(x)]))
32 #V(M2.subgraph)$name <- M2.subgraph.communities$cluster
33 #ln1 <- order(order(M2.subgraph.communities$degree, decreasing=T))
34
35 #M2.subgraph.communities <- walktrap.community(M2.subgraph, weights = E(M2.subgraph)$weight)
36
37 #
38
39 x <- which.max(sizes(M2.subgraph.communities))
40 subg1 <- induced.subgraph(M2.subgraph, which(membership(M2.subgraph.communities) == x)) #_____ LARGEST COMPONENT
41 vcount(subg1)
42 y <- which.min(sizes(M2.subgraph.communities))
43 subg2 <- induced.subgraph(M2.subgraph, which(membership(M2.subgraph.communities) == y)) #_____ SMALLEST COMPONENT
44 vcount(subg2)
45 #modularity(as.undirected(M2.subgraph), membership = cutat(M2.subgraph.communities, no = 10), #_____ MODULARITY
46 # weights = E(M2.subgraph)$weight)
47
48 #M2.subgraph.communities <- walktrap.community(M2.subgraph) #_____ OTHER HIERARCHICAL METHODS
49 #modularity(M2.subgraph.communities)
50 #M2.subgraph.communities <- spinglass.community(M2.subgraph)
51 #modularity(M2.subgraph.communities)
52 #M2.subgraph.communities <- fastgreedy.community(M2.subgraph)
53 #modularity(M2.subgraph.communities)
54 #_____ _OR
55 #sg <- M2.subgraph %>% cluster_spinglass(spins = 10) # produces 10 communities using spinglass algorithm
56 #walk <- M2.subgraph %>% cluster_walktrap() %>% cut_at(no = 10)
57 #eb <- M2.subgraph %>% cluster_edge_betweenness() %>% cut_at(no = 10)
58
59 M2.subgraph.clustering <- make_clusters(as.undirected(M2.subgraph), membership = M2.subgraph.communities$membership)
60
61 #M2.subgraph.comm_den <- as.dendrogram(M2.subgraph.communities) #_____ DENDROGRAM
62 #plot_dendrogram(M2.subgraph.comm_den)
63 #cut_at(M2.subgraph.communities, no = 2)
64 #for(i in unique(M2.subgraph.communities$membership)) {
65 # V(M2.subgraph)[Faction == i]$shape <- M2.subgraph.communities$membership + 1
66 #set.seed(1234)
67 #plot(M2.subgraph.communities, M2.subgraph)
68
69 V(M2.subgraph)$color <- M2.subgraph.communities$membership + 1
70 set.seed(1234)
71 par(mar=c(0,0,0,0))
72 plot(M2.subgraph.clustering, M2.subgraph, vertex.size = 2, vertex.frame.color = NULL, layout=M2_LGL, asp=FALSE,
73 vertex.label=NA, edge.curved=TRUE, edge.arrow.size=0.1, vertex.label.dist = 0.5, vertex.label.cex = 0.7, edge.width = 0.5)
74
75 set.seed(1234)
76 par(mar=c(0,0,0,0))
77 plot(M2.subgraph, vertex.size = 2, vertex.frame.color = NULL, layout=M2_LGL, asp=FALSE,
78 vertex.label=NA, edge.curved=TRUE, edge.arrow.size=0.1, vertex.label.dist = 0.5, vertex.label.cex = 0.7, edge.width = 0.5)
79

```

```

1 %%R
2
3 set.seed(81234)

```

```

4 #L01a = layout_nicely(M2.subgraph) #_____ Simplified network plotting
5 #L01a = layout.fruchterman.reingold(M2.subgraph)
6 L01a = layout_with_lgl(M2.subgraph)
7 par(mar=c(0,0,0,0))
8 plot(M2.subgraph,layout=L01a,vertex.size = 3,asp=FALSE,edge.curved=TRUE, vertex.frame.color = NULL,edge.arrow.size=0.04,
9       vertex.color=rainbow(max(M2.subgraph.communities$membership), alpha=0.4)[M2.subgraph.communities$membership], vertex.l
10
11 write_graph(M2.subgraph,"/content/drive/My Drive/datasets/M2.subgraph.txt",format = "graphml")
12
13 colrs <- adjustcolor( c("gray50", "tomato", "gold", "yellowgreen", "orange", "lightsteelblue2", "dark red"), alpha=.6)
14
15 #_____ Hub scores
16 #_____ # (for directed graphs)
17
18 #M2.subgraph.communities$Hub <- igraph::hub.score(M2.subgraph)$vector[M2.subgraph.communities$names]
19 #M2.subgraph.communities$cluster <- unname(ave(M2.subgraph.communities$Hub,
20 #      M2.subgraph.communities$membership,
21 #      FUN=function(x)names(x)[which.max(x)]))
22 #V(M2.subgraph)$name <- M2.subgraph.communities$cluster
23 #ctlGroups = igraph::simplify(contract(M2.subgraph, M2.subgraph.communities$membership,
24 #      vertex.attr.comb = function(x)x[1]),remove.loops=TRUE)
25 #V(ctlGroups)$hub.score <- unname(igraph::hub.score(ctlGroups)$vector)
26 ##duplicated(V(ctlGroups)$name) | duplicated(V(ctlGroups)$name, fromLast = TRUE)
27
28 #set.seed(1234)
29 #GLOa = layout_with_lgl(ctlGroups)
30 #par(mar=c(0,0,2,0))
31 #plot(ctlGroups, layout = GLOa,vertex.size = 7 * sqrt(V(ctlGroups)$hub.score), vertex.color=colrs[1],
32 #      edge.arrow.size=0.04,asp=FALSE, main="Top degree nodes",
33 #      edge.curved=TRUE, vertex.frame.color = NULL, vertex.label.cex = 0.8 * (V(ctlGroups)$hub.score),
34 #      vertex.label = ifelse(unname(igraph::hub.score(ctlGroups)$vector) > 0.85*max(unname(igraph::hub.score(ctlGroups)$vec
35 #      vertex.label.dist=0,vertex.label.family="Helvetica",vertex.label.color="black")
36
37 #_____ Degree
38 #M2.subgraph.communities <- edge.betweenness.community(as.undirected(M2.subgraph),
39 #      weights = NULL, modularity = TRUE, membership = TRUE) ##>% cut_at(no = 23)
40 #M2.subgraph.communities$degree <- (igraph::degree(M2.subgraph)[M2.subgraph.communities$names])
41 #M2.subgraph.communities$cluster <- unname(ave(M2.subgraph.communities$degree,
42 #      M2.subgraph.communities$membership,
43 #      FUN=function(x)names(x)[which.max(x)]))
44 #V(M2.subgraph)$name <- M2.subgraph.communities$cluster
45 #ctlGroups = igraph::simplify(contract(M2.subgraph, M2.subgraph.communities$membership,
46 #      vertex.attr.comb = function(x)x[1]),remove.loops=TRUE)
47 #V(ctlGroups)$degree <- unname(igraph::degree(ctlGroups))
48 ##duplicated(V(ctlGroups)$name) | duplicated(V(ctlGroups)$name, fromLast = TRUE)
49
50 set.seed(81234)
51 GLOa = layout_with_lgl(ctlGroups)
52 par(mar=c(0,0,2,0))
53 plot(ctlGroups, layout = GLOa,vertex.size = sqrt(V(ctlGroups)$degree), vertex.color=colrs[2],
54       edge.arrow.size=0.04,asp=FALSE, main="Top Degree nodes",
55       edge.curved=TRUE, vertex.frame.color = NULL, vertex.label.cex = 0.03 * (V(ctlGroups)$degree),
56       vertex.label = ifelse(unname(igraph::degree(ctlGroups)) > 0.01*max(unname(igraph::degree(ctlGroups))), V(ctlGroups)$r
57       vertex.label.dist=0,vertex.label.family="Helvetica",
58       vertex.label.color="black")
59 #_____ EigenVector
60 #M2.subgraph.communities <- edge.betweenness.community(as.undirected(M2.subgraph),
61 #      weights = NULL, modularity = TRUE, membership = TRUE) ##>% cut_at(no = 23)
62 #M2.subgraph.communities$evcent <- (igraph::evcent(M2.subgraph)$vector[M2.subgraph.communities$names])
63 #M2.subgraph.communities$cluster <- unname(ave(M2.subgraph.communities$evcent,
64 #      M2.subgraph.communities$membership,
65 #      FUN=function(x)names(x)[which.max(x)]))
66 #V(M2.subgraph)$name <- M2.subgraph.communities$cluster
67 #ctlGroups = igraph::simplify(contract(M2.subgraph, M2.subgraph.communities$membership,
68 #      vertex.attr.comb = function(x)x[1]),remove.loops=TRUE)
69 #V(ctlGroups)$evcent <- unname(igraph::evcent(ctlGroups)$vector)
70 ##duplicated(V(ctlGroups)$name) | duplicated(V(ctlGroups)$name, fromLast = TRUE)
71
72 set.seed(81234)
73 GLOa = layout_with_lgl(ctlGroups)
74 par(mar=c(0,0,2,0))
75 plot(ctlGroups, layout = GLOa,vertex.size = 7 * sqrt(V(ctlGroups)$evcent), vertex.color=colrs[3],
76       edge.arrow.size=0.04,asp=FALSE, main="Top Eigenvector nodes",
77       edge.curved=TRUE, vertex.frame.color = NULL, vertex.label.cex = 0.8 * (V(ctlGroups)$evcent),
78       vertex.label = ifelse(unname(igraph::evcent(ctlGroups)$vector) > 0.85*max(unname(igraph::evcent(ctlGroups)$vector))),
79       vertex.label.dist=0,vertex.label.family="Helvetica",
80       vertex.label.color="black")

```

```

81 # _____ Closeness
82 M2.subgraph.communities <- edge.betweenness.community(as.undirected(M2.subgraph),
83 # weights = NULL, modularity = TRUE, membership = TRUE) #>% cut_at(no = 23)
84 M2.subgraph.communities$closeness <- (igraph::closeness(M2.subgraph)[M2.subgraph.communities$names])
85 M2.subgraph.communities$cluster <- unname(ave(M2.subgraph.communities$closeness,
86 M2.subgraph.communities$membership,
87 FUN=function(x)names(x)[which.max(x)]))
88 V(M2.subgraph)$name <- M2.subgraph.communities$cluster
89 ctlGroups = igraph::simplify(contract(M2.subgraph, M2.subgraph.communities$membership,
90 vertex.attr.comb = function(x)x[1]),remove.loops=TRUE)
91 V(ctlGroups)$closeness <- unname(igraph::closeness(ctlGroups))
92 ##duplicated(V(ctlGroups)$name) | duplicated(V(ctlGroups)$name, fromLast = TRUE)
93
94 set.seed(81234)
95 GLOa = layout_with_lgl(ctlGroups)
96 par(mar=c(0,0,2,0))
97 plot(ctlGroups, layout = GLOa,vertex.size = 1500 * V(ctlGroups)$closeness, vertex.color=colrs[4],
98 edge.arrow.size=0.04,asp=FALSE, main="Top Closeness nodes",
99 edge.curved=TRUE, vertex.frame.color = NULL, vertex.label.cex = 150 * V(ctlGroups)$closeness,
100 vertex.label = ifelse(unname(igraph::closeness(ctlGroups)) > 0.8*max(unname(igraph::closeness(ctlGroups))), V(ctlGrou
101 vertex.label.dist=0,vertex.label.family="Helvetica",
102 vertex.label.color="black")
103 # _____ Betweenness
104 M2.subgraph.communities <- edge.betweenness.community(as.undirected(M2.subgraph),
105 # weights = NULL, modularity = TRUE, membership = TRUE) #>% cut_at(no = 23)
106 M2.subgraph.communities$betweenness <- (igraph::betweenness(M2.subgraph)[M2.subgraph.communities$names])
107 M2.subgraph.communities$cluster <- unname(ave(M2.subgraph.communities$betweenness,
108 M2.subgraph.communities$membership,
109 FUN=function(x)names(x)[which.max(x)]))
110 V(M2.subgraph)$name <- M2.subgraph.communities$cluster
111 ctlGroups = igraph::simplify(contract(M2.subgraph, M2.subgraph.communities$membership,
112 vertex.attr.comb = function(x)x[1]),remove.loops=TRUE)
113 V(ctlGroups)$betweenness <- unname(igraph::betweenness(ctlGroups))
114 ##duplicated(V(ctlGroups)$name) | duplicated(V(ctlGroups)$name, fromLast = TRUE)
115
116 set.seed(81234)
117 GLOa = layout_with_lgl(ctlGroups)
118 par(mar=c(0,0,2,0))
119 plot(ctlGroups, layout = GLOa,vertex.size = 0.1 * sqrt(V(ctlGroups)$betweenness), vertex.color=colrs[6],
120 edge.arrow.size=0.04,asp=FALSE, main="Top Betweenness nodes",
121 edge.curved=TRUE, vertex.frame.color = NULL, vertex.label.cex = 0.00045 * (V(ctlGroups)$betweenness),
122 vertex.label = ifelse(unname(igraph::betweenness(ctlGroups)) > 0.7*max(unname(igraph::betweenness(ctlGroups))), V(ctl
123 vertex.label.dist=0,vertex.label.family="Helvetica",
124 vertex.label.color="black")
125 # _____ Authority
126 ##M2.subgraph.communities <- edge.betweenness.community(as.undirected(M2.subgraph), # (for directed graphs)
127 ## weights = NULL, modularity = TRUE, membership = TRUE) #>% cut_at(no = 23)
128 M2.subgraph.communities$Authority <- igraph::authority.score(M2.subgraph)$vector[M2.subgraph.communities$names]
129 M2.subgraph.communities$cluster <- unname(ave(M2.subgraph.communities$Authority,
130 M2.subgraph.communities$membership,
131 FUN=function(x)names(x)[which.max(x)]))
132 #V(M2.subgraph)$name <- M2.subgraph.communities$cluster
133 #ctlGroups = igraph::simplify(contract(M2.subgraph, M2.subgraph.communities$membership,
134 # vertex.attr.comb = function(x)x[1]),remove.loops=TRUE)
135 #V(ctlGroups)$authority.score <- unname(igraph::authority.score(ctlGroups)$vector)
136 ##duplicated(V(ctlGroups)$name) | duplicated(V(ctlGroups)$name, fromLast = TRUE)
137
138 #set.seed(1234)
139 #GLOa = layout_with_lgl(ctlGroups)
140 #par(mar=c(0,0,0,0))
141 #plot(ctlGroups, layout = GLOa,vertex.size = 7 * sqrt(V(ctlGroups)$authority.score), vertex.color=colrs[7],
142 # edge.arrow.size=0.04,asp=FALSE, main="Community number with the highest degree node",
143 # edge.curved=TRUE, vertex.frame.color = NULL, vertex.label.cex = 0.8 * (V(ctlGroups)$authority.score),
144 # vertex.label = ifelse(unname(igraph::authority.score(ctlGroups)$vector) > 0.85*max(unname(igraph::authority.score(ct
145 # vertex.label.dist=0,vertex.label.family="Helvetica",
146 # vertex.label.color="black")
147
148 # _____ Community numbers on graph
149
150 M2.subgraph.communities <- edge.betweenness.community(as.undirected(M2.subgraph),
151 weights = NULL, modularity = TRUE, membership = TRUE) #>% cut_at(no = 23)
152 M2.subgraph.communities$Hub <- igraph::hub.score(M2.subgraph)$vector[M2.subgraph.communities$names]
153 M2.subgraph.communities$cluster <- unname(ave(M2.subgraph.communities$Hub,
154 M2.subgraph.communities$membership,
155 FUN=function(x)names(x)[which.max(x)]))
156 V(M2.subgraph)$name <- M2.subgraph.communities$membership[which(M2.subgraph.communities$names == M2.subgraph.communities$
157 ctlGroups = igraph::simplify(contract(M2.subgraph, M2.subgraph.communities$membership,

```

```

158             vertex.attr.comb = function(x)x[1]),remove.loops=TRUE)
159 V(ctlGroups)$hub.score <- unname(igraph::hub.score(ctlGroups)$vector)
160
161 set.seed(81234)
162 GLOa = layout_with_lgl(ctlGroups)
163 par(mar=c(0,0,0,0))
164 plot(ctlGroups, layout = GLOa,vertex.size = 5,
165       edge.arrow.size=0.04,asp=FALSE, main="Community numbers",
166       edge.curved=TRUE, vertex.frame.color = NULL, vertex.label.cex = 1.2,vertex.color=colrs[5],
167       vertex.label = ifelse(unname(igraph::hub.score(ctlGroups)$vector) >= 0, V(ctlGroups)$name, NA),
168       vertex.label.dist=0.5,vertex.label.family="Helvetica",vertex.label.color="black")
169
170 M2.subgraph <- read_graph("/content/drive/My Drive/datasets/M2.subgraph.txt",format = "graphml")
171 subg1 <- induced.subgraph(M2.subgraph, which(membership(M2.subgraph.communities) == 67))
172 V(subg1)$name
173 #
174 Rada = min(dist(GLOa))
175 L02a = matrix(0, nrow=vcount(M2.subgraph), ncol=2)
176 for(i in unique(M2.subgraph.communities$membership)) {
177   #vids <- M2.subgraph.communities$membership[[i]]
178   #Clusta = induced_subgraph(M2.subgraph, vids)
179   Clusta = induced_subgraph(M2.subgraph, which(M2.subgraph.communities$membership == i))
180   CLOa = layout_with_fr(Clusta)
181   L02a[which(M2.subgraph.communities$membership == i), ] = t(t(scale(CLOa)*Rada/3) + GLOa[i,])
182 }
183
184 set.seed(81234)
185 par(mar=c(0,0,0,0))
186 plot(M2.subgraph, layout=L02a, vertex.label=NA, vertex.size = 3, edge.arrow.size=0.04, asp=FALSE,edge.curved=TRUE, vertex.
187       vertex.color=rainbow(max(M2.subgraph.communities$membership), alpha=0.4)[M2.subgraph.communities$membership])
188

```

```

1 %%R -o commSummary2
2
3 #install.packages("centiserve")
4 #library(centiserve)
5
6 #arpack_defaults$maxiter = 10000
7
8 #Hub <- igraph::hub.score(M2.subgraph, options=arpack_defaults)$vector
9 #Authority <- igraph::authority.score(M2.subgraph)
10 Closeness <- igraph::closeness(M2.subgraph)
11 Betweenness <- igraph::betweenness(M2.subgraph)
12 #centr_lap_top <- centiserve::laplacian(M2.subgraph)
13 Degree <- igraph::degree(M2.subgraph)
14 EigVec <- igraph::eigvec(M2.subgraph)$vector
15
16 commSummary <- data.frame(
17   M2.subgraph.communities$names,
18   M2.subgraph.communities$membership,
19   #M2.subgraph.communities$modularity,
20   #centr_lap_top,
21   #Hub,
22   #Authority,
23   Closeness,
24   Betweenness,
25   Degree,
26   EigVec
27 )
28
29 colnames(commSummary) <- c("Gene", "Community", "Closeness", "Betweenness", "Degree", "EigVec")
30 options(scipen=999)
31
32 #save(commSummary, file="/content/drive/My Drive/datasets/commSummary.Rdata")
33
34 #
35
36 #load("/content/drive/My Drive/datasets/commSummary.Rdata")
37
38 cols1 <- grep("Closeness", names(commSummary))
39 #commSummary2 <- commSummary[rowSums(commSummary[cols1] > 1.3*mean(centr_lap_top)) == length(cols1), ]
40 #commSummary2 <- commSummary[rowSums(commSummary['Hub_Score'] > 1.5*mean(Hub))== length(cols1), ]
41 commSummary2 <- commSummary[rowSums(commSummary['Closeness'] > min(Closeness))== length(cols1), ]
42 #commSummary2 <- commSummary[rowSums(commSummary[cols1] >= min(centr_lap_top)) == length(cols1), ]
43

```

```
44 head(commSummary2)
45
```

```
1 import numpy as np
2 import pandas as pd
3
4 Gene_Lap = commSummary2.groupby('Community').count().sort_values(by=['Gene'], ascending=False)
5 comm_to_R=[]
6
7 #for i in range(len(list(Gene_Lap.index)[0:4])):      #_____ Top 5 communities
8 for i in range(len(list(Gene_Lap.index))):            #_____ All communities
9     comm_to_R.append(list(Gene_Lap.index)[i])
10
11 back_genes = pd.read_csv('/content/drive/My Drive/datasets/back_genes.csv')
12 background_uni = back_genes['x']
13 background_uni.dropna(how='all', inplace=True)
14
```

```
1 Start coding or generate with AI.
```

```
1 Start coding or generate with AI.
```

```
1 Start coding or generate with AI.
```

```
1 %%R -i comm_to_R -o commSum2_subg_to
2
3 inp_ab = 1.3
4 commSum2_subg_to <- list()
5 con_fin <- list()
6
7 for(i in 1:length(comm_to_R)) {
8   subg <- paste("subg", i, sep = "")
9   subg_to<-assign(subg, induced.subgraph(M2.subgraph, which(membership(M2.subgraph.communities) == unlist(comm_to_R)[i])))
10
11 #subg3_lap_top <- centiserve::laplacian(subg_to)
12 #subg3_hub_top <- igraph::hub.score(subg_to, options=arpack_defaults)$vector
13 subg3_clos_top <- igraph::closeness(subg_to)
14 subg3_betw_top <- igraph::betweenness(subg_to)
15 subg3_deg_top <- igraph::degree(subg_to)
16 subg3_eig_top <- igraph::evcent(subg_to)$vector
17
18 commSum_subg3 <- data.frame(
19   V(subg_to)$name,
20   subg3_clos_top,
21   subg3_betw_top,
22   subg3_deg_top,
23   subg3_eig_top
24 )
25
26 colnames(commSum_subg3) <- c("Gene", "Closeness", "Betweenness", "Degree", "EigenVector")
27
28 options(scipen=999)
29
30 cols3 <- grep("Closeness", names(commSum_subg3))
31
32 commSum2_subg <- paste("commSum2_subg", i, sep = "")
33 commSum2_subg_to[i] <- list(assign(commSum2_subg, commSum_subg3[rowSums(commSum_subg3[cols3] >= min(subg3_clos_top)) == 1e
34 })
35
```

```
1 '''
2 import numpy
3 import numpy as np
4 import pandas as pd
5
6 con_fin1 = []
7 fin_tmpa = []
8 fin_tmpb = []
9 fin_tmpc = []
10 fin_tmpd = []
11
12 for i in range(len(commSum2_subg_to)) :
13     con_fin = []
```

```

14 commSum2_subg_tmp1 = pd.DataFrame(commSum2_subg_to[i])
15
16 commSum2_subg_tmp2a = pd.DataFrame(np.transpose(commSum2_subg_tmp1.values[1,:]))
17 commSum2_subg_tmp2b = pd.DataFrame(np.transpose(commSum2_subg_tmp1.values[2,:]))
18 commSum2_subg_tmp2c = pd.DataFrame(np.transpose(commSum2_subg_tmp1.values[3,:]))
19 commSum2_subg_tmp2d = pd.DataFrame(np.transpose(commSum2_subg_tmp1.values[4,:]))
20
21 commSum2_subg_tmp2a.columns = ['Closeness']
22 commSum2_subg_tmp2b.columns = ['Betweenness']
23 commSum2_subg_tmp2c.columns = ['Degree']
24 commSum2_subg_tmp2d.columns = ['Eigenvector']
25
26 commSum2_subg_tmp1 = pd.DataFrame(np.transpose(commSum2_subg_tmp1.values[0,:]))
27 commSum2_subg_tmp1.columns = ['Gene']
28
29 commSum2_subg_tmp = pd.concat([commSum2_subg_tmp1.reset_index(drop=True),
30                               commSum2_subg_tmp2a, commSum2_subg_tmp2b,
31                               commSum2_subg_tmp2c, commSum2_subg_tmp2d], axis=1)
32
33 commSum2_subg_tmp['Closeness'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*0.3)).reset_index()
34 fin_tmpa.append(commSum2_subg_tmp['Closeness'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*0.3)).reset_index())
35 commSum2_subg_tmp['Betweenness'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*0.3)).reset_index()
36 fin_tmpb.append(commSum2_subg_tmp['Betweenness'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*0.3)).reset_index())
37 commSum2_subg_tmp['Degree'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*0.3)).reset_index()
38 fin_tmpc.append(commSum2_subg_tmp['Degree'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*0.3)).reset_index())
39 commSum2_subg_tmp['Eigenvector'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*0.3)).reset_index()
40 fin_tmpd.append(commSum2_subg_tmp['Eigenvector'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*0.3)).reset_index())
41
42 con_fin.append(commSum2_subg_tmp1.values[fin_tmpa[i]]['index'])
43 con_fin.append(commSum2_subg_tmp1.values[fin_tmpb[i]]['index'])
44 con_fin.append(commSum2_subg_tmp1.values[fin_tmpc[i]]['index'])
45 con_fin.append(commSum2_subg_tmp1.values[fin_tmpd[i]]['index'])
46 flat_con_fin = [item for sublist in con_fin for item in sublist]
47 con_finx = [i[0] for i in flat_con_fin]
48 con_fin1.append(list(set(con_finx)))
49
50 %R -i con_fin1
51 #%R save(con_fin1, file="/content/drive/My Drive/datasets/con_fin1.Rdata")
52
53 '''
54
55 '''
56 You can choose genes by 'Closeness', 'Eigenvector', 'Degree', or 'Betweenness'
57 or all of them (as in this module of the comment). You can also disjoint the
58 biggest communities into smaller ones and try the genes from each one of them!!
59 '''

```

```

1 '''
2 %R -i con_fin1
3
4 ensLookup_tmp = list()
5
6 for (i in 1:length(con_fin1)) {
7   ensLookup_tmp[i] <- unlist(con_fin1[i])
8 }
9 print("top 30% of genes kept:") # nlargest(int(commSum2_subg_tmp.shape[0]*0.3))
10
11 print("Number of communities:")
12 print(length(ensLookup_tmp))
13
14 j=0
15 for (i in 1:length(ensLookup_tmp)) {
16   if (length(unlist(con_fin1[i]))>2){
17     j=j+1
18   }
19 }
20 print("Number of communities with more than 2 genes:")
21 print(j)
22
23 '''
24

```

```

1 # _____ Installing li
2 %R
3
4 if (!requireNamespace("BiocManager", quietly = TRUE))

```

```

5   install.packages("BiocManager")
6   BiocManager::install("biomaRt", force = TRUE)
7   library(biomaRt)
8   #require("biomaRt")
9   mart <- useMart("ENSEMBL_MART_ENSEMBL")
10  mart <- useDataset("hsapiens_gene_ensembl", mart)
11
12  devtools::install_github(c("GuangchuangYu/DOSE", "GuangchuangYu/enrichplot", "GuangchuangYu/GOSemSim", "eliocamp/ggnewscale
13
14  BiocManager::install("clusterProfiler")
15  library(clusterProfiler)
16
17  organism = "org.Hs.eg.db"
18  BiocManager::install(organism, character.only = TRUE)
19  library(organism, character.only = TRUE)
20
21  BiocManager::install("DOSE")
22  library(DOSE)
23
24  #BiocManager::install("ReactomePA")
25  #library(ReactomePA)
26

```

```

1  '''
2  %%R -i con_fin,background_uni
3
4  # _____ I. GO (ENTREZ through biomaRt)
5                                     #(Plot communities one-by-one, manually)
6  df = as.data.frame(org.Hs.egGO)
7  go_gene_list = unique(sort(df$gene_id))
8
9  require("biomaRt")
10 mart <- useMart("ENSEMBL_MART_ENSEMBL")
11 mart <- useDataset("hsapiens_gene_ensembl", mart)
12
13 ensLookup_1 = list()
14 annotLookup = list()
15
16 geneList <- background_uni
17 geneList <- unlist(geneList)
18 geneList <- geneList[order(nchar(geneList), geneList)]
19
20 df = as.data.frame(org.Hs.egGO)
21 go_gene_list = unique(sort(df$gene_id))
22 go_gene_list.df <- bitr(go_gene_list, fromType = "ENTREZID",
23                         toType = c("ENSEMBL", "SYMBOL"),
24                         OrgDb = org.Hs.eg.db)
25
26 geneList.df <- bitr(as.character(unique(sort(as.character(geneList))))), fromType = "ENTREZID",
27                    toType = c("ENSEMBL", "SYMBOL"),
28                    OrgDb = org.Hs.eg.db)
29
30 dfk = as.data.frame(org.Hs.egPATH)
31 kegg_gene_list = unique(sort(dfk$gene_id))
32
33 xp = list()
34 ego1 = list()
35 ego2 = list()
36 ego3 = list()
37 x = list()
38 kk = list()
39 gg01 = list()
40 gg02 = list()
41 gene.df = list()
42
43 #if (file.exists(fn)) {
44 #   file.remove(fn)
45 #}
46
47 for (i in 1:length(con_fin1)) {
48   ensLookup_1[[i]] <- unlist(con_fin1[i])
49
50   if (length(unlist(con_fin1[i]))>2){
51     annotLookup[[i]] <- getBM(
52       mart=mart,
53       attributes=c("ensembl_transcript_id", "ensembl_gene_id",

```

```

54   "gene_biotype", "external_gene_name", "entrezgene_id"),
55   filter="ensembl_gene_id",
56   values=ensLookup_1[[i]],
57   uniqueRows=TRUE)
58   annotLookup[[i]] <- data.frame(ensLookup_1[[i]][match(annotLookup[[i]]$ensembl_gene_id, ensLookup_1[[i])]),
59   annotLookup[[i]])
60   annotLookup[[i]] <- subset(annotLookup[[i]], (!is.na(annotLookup[[i]]['entrezgene_id'])))
61   annotLookup[[i]] <- annotLookup[[i]][annotLookup[[i]]['gene_biotype'] == "protein_coding", ]
62 }
63 #else {annotLookup[[i]] == 0
64 #   }
65 #colnames(annotLookup[[i]]) <- c(
66 #   "original_id",
67 #   c("ensembl_transcript_id", "ensembl_gene_id",
68 #     "gene_biotype", "external_gene_name", "EntrezID"))
69
70 #_____ Using bitr for translating between "ENSEM
71 }
72
73 for(i in 1:length(annotLookup)) {
74
75   print("_____ Ia. enrichGO (BP) _____")
76   l1 <- enrichGO(gene       = unique(sort(annotLookup[[i]]$ensembl_gene_id)),
77                 universe    = go_gene_list.df$ENSEMBL,                #org.Hs.egGO
78                 #universe    = geneList.df$ENSEMBL,                  #background genes
79                 OrgDb        = org.Hs.eg.db,
80                 #keyType     = "ENTREZID",
81                 keyType      = "ENSEMBL",
82                 ont          = "BP",
83                 pAdjustMethod = "BH",                                  #pAdjustMethod = "fdr"
84                 pvalueCutoff = 0.05,
85                 qvalueCutoff = 0.1,
86                 readable     = TRUE)
87   if (length(l1$ID) == 0) {
88     ego1[[i]] <- "NULL"
89   }
90   else{
91     ego1[[i]] <- data.frame(
92       i,
93       l1$ID,
94       l1$Description,
95       l1$GeneRatio,
96       l1$BgRatio,
97       l1$pvalue,
98       l1$p.adjust,
99       l1$qvalue,
100      l1$geneID,
101      l1$Count
102    )
103    colnames(ego1[[i]]) <- c("Community#", "ID", "Description", "GeneRatio",
104      "BgRatio", "pvalue", "p.adjust", "qvalue",
105      "geneID", "Count")
106    options(scipen=999)
107  }
108
109   print("_____ Ib. enrichGO (CC) _____")
110   l2 <- enrichGO(gene       = unique(sort(annotLookup[[i]]$ensembl_gene_id)),
111                 universe    = go_gene_list.df$ENSEMBL,                #org.Hs.egGO
112                 #universe    = geneList.df$ENSEMBL,                  #background genes
113                 OrgDb        = org.Hs.eg.db,
114                 #keyType     = "ENTREZID",
115                 keyType      = "ENSEMBL",
116                 ont          = "CC",
117                 pAdjustMethod = "BH",                                  #pAdjustMethod = "fdr"
118                 pvalueCutoff = 0.05,
119                 qvalueCutoff = 0.1,
120                 readable     = TRUE)
121   if (length(l2) == 0){
122     ego2[[i]] <- "NULL"
123   }
124   else{
125     ego2[[i]] <- l2
126   }
127   #head(summary(as.data.frame(ego2[[i]])))
128   #print(ego2[[i]])
129   cat('\n')
130   print("_____ Ic. enrichGO (MF) _____")

```

```

131 l3 <- enrichGO(gene      = unique(sort(annotLookup[[i]]$ensembl_gene_id)),
132                universe   = go_gene_list.df$ENSEMBL,                #org.Hs.egGO
133                #universe   = geneList.df$ENSEMBL,                    #background genes
134                OrgDb       = org.Hs.eg.db,
135                #keyType    = "ENTREZID",
136                keyType     = "ENSEMBL",
137                ont         = "MF",
138                pAdjustMethod = "BH",                                #pAdjustMethod = "fdr"
139                pvalueCutoff = 0.05,
140                qvalueCutoff = 0.1,
141                readable     = TRUE)
142 if (length(l3) == 0){
143   ego3[[i]] <- "NULL"
144 }
145 else{
146   ego3[[i]] <- l3
147 }
148 #head(summary(as.data.frame(ego3[[i]])))
149 #print(ego3[[i]])
150 cat('\n')
151 print("_____II. enrichDO_____")
152 l4 <- enrichDO(gene      = as.character(unique(sort(annotLookup[[i]]$entrezgene_id))),
153               ont        = "DO",
154               pvalueCutoff = 0.05,
155               pAdjustMethod = "BH",
156               universe    = go_gene_list,                #org.Hs.egGO
157               #universe   = geneList.df$ENTREZID,        #background genes
158               minGSSize   = 10,
159               maxGSSize   = 1000,
160               qvalueCutoff = 0.1,
161               readable     = TRUE)
162 if (length(l4$ID) == 0) {
163   x[[i]] <- "NULL"
164 }
165 else{
166   x[[i]] <- data.frame(
167     i,
168     l4$ID,
169     l4$Description,
170     l4$GeneRatio,
171     l4$BgRatio,
172     l4$pvalue,
173     l4$p.adjust,
174     l4$qvalue,
175     l4$geneID,
176     l4$Count
177   )
178   colnames(x[[i]]) <- c("Community#", "ID", "Description", "GeneRatio",
179                        "BgRatio", "pvalue", "p.adjust", "qvalue",
180                        "geneID", "Count")
181   options(scipen=999)
182 }
183
184 print("_____III. enrichKEGG_____")
185 l5 <- enrichKEGG(gene      = as.character(unique(sort(annotLookup[[i]]$entrezgene_id))),
186                 organism   = 'hsa',
187                 pvalueCutoff = 0.05,
188                 universe    = kegg_gene_list,            #org.Hs.egPATH
189                 #universe   = geneList.df$ENTREZID,      #background genes
190                 qvalueCutoff = 0.1)
191 if (length(l5$ID) == 0) {
192   kk[[i]] <- "NULL"
193 }
194 else{
195   kk[[i]] <- data.frame(
196     i,
197     l5$ID,
198     l5$Description,
199     l5$GeneRatio,
200     l5$BgRatio,
201     l5$pvalue,
202     l5$p.adjust,
203     l5$qvalue,
204     l5$geneID,
205     l5$Count
206   )
207   colnames(kk[[i]]) <- c("Community#", "ID", "Description", "GeneRatio",

```

```

208         "BgRatio", "pvalue", "p.adjust", "qvalue",
209         "geneID", "Count")
210 options(scipen=999)
211 }
212 cat('\n')
213 cat('\n')
214 cat('\n')
215 cat('\n')
216 cat('\n')
217 print("#####")
218 }
219 '''

```

```

1 '''
2 # _____ Print all of the above results to Tables
3 %R -o kk,x,ego1
4
5
6 for i in range(len(ego1)):
7     if (len(ego1[i])!=1):
8         df1 = pd.DataFrame(ego1[i]).T
9         df1.rename(columns={0: 'Community#', 1: 'ID',
10             2: 'Description', 3: 'GeneRatio',
11             4: 'BgRatio', 5: 'pvalue',
12             6: 'p.adjust', 7: 'qvalue',
13             8: 'geneID', 9: 'Count'},inplace=True)
14         print(df1.to_string())
15         print("#####")
16 print("_____")
17 print("_____")
18 print("_____")
19
20
21 for i in range(len(x)):
22     if (len(x[i])!=1):
23         df4 = pd.DataFrame(x[i]).T
24         df4.rename(columns={0: 'Community#', 1: 'ID',
25             2: 'Description', 3: 'GeneRatio',
26             4: 'BgRatio', 5: 'pvalue',
27             6: 'p.adjust', 7: 'qvalue',
28             8: 'geneID', 9: 'Count'},inplace=True)
29         print(df4.to_string())
30         print("#####")
31 print("_____")
32 print("_____")
33 print("_____")
34
35
36 for i in range(len(kk)):
37     if (len(kk[i])!=1):
38         df5 = pd.DataFrame(kk[i]).T
39         df5.rename(columns={0: 'Community#', 1: 'ID',
40             2: 'Description', 3: 'GeneRatio',
41             4: 'BgRatio', 5: 'pvalue',
42             6: 'p.adjust', 7: 'qvalue',
43             8: 'geneID', 9: 'Count'},inplace=True)
44         print(df5.to_string())
45         print("#####")
46 '''

```

```

1
2 fin_tmpar = []
3 fin_tmpr = []
4 fin_tmprc = []
5 fin_tmprd = []
6 con_fin1t = []
7
8 for i in range(18) :
9     con_fin = []
10     commSum2_subg_tmp1 = pd.DataFrame(commSum2_subg_to[i])
11
12     commSum2_subg_tmp2a = pd.DataFrame(np.transpose(commSum2_subg_tmp1.values[1,:]))
13     commSum2_subg_tmp2b = pd.DataFrame(np.transpose(commSum2_subg_tmp1.values[2,:]))
14     commSum2_subg_tmp2c = pd.DataFrame(np.transpose(commSum2_subg_tmp1.values[3,:]))
15     commSum2_subg_tmp2d = pd.DataFrame(np.transpose(commSum2_subg_tmp1.values[4,:]))
16

```

```

17 commSum2_subg_tmp2a.columns = ['Closeness']
18 commSum2_subg_tmp2b.columns = ['Betweenness']
19 commSum2_subg_tmp2c.columns = ['Degree']
20 commSum2_subg_tmp2d.columns = ['EigenVector']
21
22 commSum2_subg_tmp1 = pd.DataFrame(np.transpose(commSum2_subg_tmp1.values[0,:]))
23 commSum2_subg_tmp1.columns = ['Gene']
24
25 commSum2_subg_tmp = pd.concat([commSum2_subg_tmp1.reset_index(drop=True),
26                               commSum2_subg_tmp2a, commSum2_subg_tmp2b,
27                               commSum2_subg_tmp2c, commSum2_subg_tmp2d], axis=1)
28
29 fin_tmpr.append(commSum2_subg_tmp['Closeness'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*1)).reset_index())
30 fin_tmpr.append(commSum2_subg_tmp['Betweenness'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*1)).reset_index())
31 fin_tmpr.append(commSum2_subg_tmp['Degree'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*1)).reset_index())
32 fin_tmpr.append(commSum2_subg_tmp['EigenVector'].astype(float).nlargest(int(commSum2_subg_tmp.shape[0]*1)).reset_index())
33
34
35 con_fin.append(commSum2_subg_tmp1.values[fin_tmpr[i]['index']])
36 con_fin.append(commSum2_subg_tmp1.values[fin_tmpr[i]['index']])
37 con_fin.append(commSum2_subg_tmp1.values[fin_tmpr[i]['index']])
38 con_fin.append(commSum2_subg_tmp1.values[fin_tmpr[i]['index']])
39 flat_con_fin = [item for sublist in con_fin for item in sublist]
40 con_finx = [i[0] for i in flat_con_fin]
41 con_fin1t.append(list(set(con_finx)))
42 print(len(con_fin1t[i]))
43 %R -i con_fin1t
44

```

```

1 %R                                     #_____ I. EnrichGO
2
3 ego1r = list()
4 annotLookup = list()
5 ensLookup_tmpr = list()
6
7 df = as.data.frame(org.Hs.egGO)
8 go_gene_list = unique(sort(df$gene_id))
9 go_gene_list.df <- bitr(go_gene_list, fromType = "ENTREZID",
10                        toType = c("ENSEMBL", "SYMBOL"),
11                        OrgDb = org.Hs.eg.db)
12
13 for (i in 1:18) {
14   ensLookup_tmpr[[i]] <- unlist(con_fin1t[i])
15   #print(head(ensLookup_tmpr[[i]]))
16
17   if (length(unlist(con_fin1t[i]))>2){
18     annotLookup[[i]] <- getBM(
19       mart=mart,
20       attributes=c("ensembl_transcript_id", "ensembl_gene_id",
21                   "gene_biotype", "external_gene_name", "entrezgene_id"),
22       filter="ensembl_gene_id",
23       values=ensLookup_tmpr[[i]],
24       uniqueRows=TRUE)
25     annotLookup[[i]] <- data.frame(ensLookup_tmpr[[i]][match(annotLookup[[i]]$ensembl_gene_id, ensLookup_tmpr[[i]])),
26                                   annotLookup[[i]])
27     annotLookup[[i]] <- subset(annotLookup[[i]], (!is.na(annotLookup[[i]]['entrezgene_id'])))
28     annotLookup[[i]] <- annotLookup[[i]][annotLookup[[i]]['gene_biotype'] == "protein_coding", ]
29   }
30
31   print("_____Ia. enrichGO (BP) _____")
32   l1 <- enrichGO(gene       = unique(sort(ensLookup_tmpr[[i]])),
33                  universe   = go_gene_list.df$ENSEMBL,                #org.Hs.egGO
34                  #universe   = geneList.df$ENSEMBL,                  #background genes
35                  OrgDb      = org.Hs.eg.db,
36                  #keyType    = "ENTREZID",
37                  keyType    = "ENSEMBL",
38                  ont         = "BP",
39                  pAdjustMethod = "BH",                                #pAdjustMethod = "fdr"
40                  pvalueCutoff = 0.05,
41                  qvalueCutoff = 0.1,
42                  readable    = TRUE)
43   if (length(l1$ID) == 0) {
44     ego1r[[i]] <- "NULL"
45   }
46   else{
47     ego1r[[i]] <- data.frame(

```

```

48 i,
49 l1$ID,
50 l1$Description,
51 l1$GeneRatio,
52 l1$BgRatio,
53 l1$pvalue,
54 l1$p.adjust,
55 l1$qvalue,
56 l1$geneID,
57 l1$Count
58 )
59 colnames(ego1r[[i]]) <- c("Community#", "ID", "Description", "GeneRatio",
60                           "BgRatio", "pvalue", "p.adjust", "qvalue",
61                           "geneID", "Count")
62 options(scipen=999)
63 }
64 }

```

```

1 %R -o ego1r
2
3 for i in range(len(ego1r)):
4     if (len(ego1r[i])!=1):
5         df1 = pd.DataFrame(ego1r[i]).T
6         df1.rename(columns={0: 'Community#', 1: 'ID',
7                             2: 'Description', 3: 'GeneRatio',
8                             4: 'BgRatio', 5: 'pvalue',
9                             6: 'p.adjust', 7: 'qvalue',
10                            8: 'geneID', 9: 'Count'},inplace=True)
11         print(df1.to_string())
12         print("#####")
13

```

```

1 %%R                                     #_____ II. KEGG
2
3 dfk = as.data.frame(org.Hs.egPATH)
4 kegg_gene_list = unique(sort(dfk$gene_id))
5
6 ensLookup_tmpr = list()
7 kk = list()
8
9 for (i in 1:18)
10 {
11     ensLookup_tmpr[[i]] <- unlist(con_fin1t[i])
12     #print(head(ensLookup_tmpr[[i]]))
13
14     if (length(unlist(con_fin1t[i]))>2){
15         annotLookup[[i]] <- getBM(
16             mart=mart,
17             attributes=c("ensembl_transcript_id", "ensembl_gene_id",
18                         "gene_biotype", "external_gene_name", "entrezgene_id"),
19             filter="ensembl_gene_id",
20             values=ensLookup_tmpr[[i]],
21             uniqueRows=TRUE)
22         annotLookup[[i]] <- data.frame(ensLookup_tmpr[[i]][match(annotLookup[[i]]$ensembl_gene_id, ensLookup_tmpr[[i]])),
23             annotLookup[[i]])
24         annotLookup[[i]] <- subset(annotLookup[[i]], (!is.na(annotLookup[[i]]['entrezgene_id'])))
25         annotLookup[[i]] <- annotLookup[[i]][annotLookup[[i]]['gene_biotype'] == "protein_coding", ]
26     }
27
28     print("_____ III. enrichKEGG_____")
29     l5 <- enrichKEGG(gene         = as.character(unique(sort(annotLookup[[i]]$entrezgene_id))),
30                     organism      = 'hsa',
31                     pvalueCutoff = 0.05,
32                     universe      = kegg_gene_list,                                #org.Hs.egPATH
33                     #universe     = geneList.df$ENTREZID,                        #background genes
34                     qvalueCutoff = 0.1)
35     if (length(l5$ID) == 0) {
36         kk[[i]] <- "NULL"
37     }
38     else{
39         kk[[i]] <- data.frame(
40             i,
41             l5$ID,
42             l5$Description,
43             l5$GeneRatio,
44             l5$BgRatio,

```

```

45 15$pvalue,
46 15$p.adjust,
47 15$qvalue,
48 15$geneID,
49 15$Count
50 )
51 colnames(kk[[i]]) <- c("Community#", "ID", "Description", "GeneRatio",
52                        "BgRatio", "pvalue", "p.adjust", "qvalue",
53                        "geneID", "Count")
54 options(scipen=999)
55 }
56 print("#####")
57 }
58

```

```

1 %R -o kk
2
3 for i in range(len(kk)):
4     if (len(kk[i])!=1):
5         df5 = pd.DataFrame(kk[i]).T
6         df5.rename(columns={0: 'Community#', 1: 'ID',
7                             2: 'Description', 3: 'GeneRatio',
8                             4: 'BgRatio', 5: 'pvalue',
9                             6: 'p.adjust', 7: 'qvalue',
10                            8: 'geneID', 9: 'Count'},inplace=True)
11         print(df5.to_string())
12         print("#####")
13

```

```

1 %%R                                     #_____ Community numbers and how many genes they
2
3 lop1 = list()
4 lop2 = list()
5
6 for(i in 1:length(commSum2_subg_to)) {
7     subg1 <- induced.subgraph(M2.subgraph, which(membership(M2.subgraph.communities) == i))
8     #print(length(V(subg1)$name))
9     lop1[i] <- length(V(subg1)$name)
10    lop2[i] <- i
11 }
12
13 gene_number = c(unlist(lop1))
14 community_number = c(unlist(lop2))
15 df = data.frame(gene_number, community_number)
16 head(df[order(df$gene_number, decreasing = TRUE),],100 )
17

```

```

1 %%R
2
3 #_____
4 #for(i in 1:10) {
5     #subg1 <- induced.subgraph(M2.subgraph, which(membership(M2.subgraph.communities) == i))
6     #print(length(V(subg1)$name))
7 }
8
9     #number of genes      #community
10 #[1]      741             1
11 #[1]       6             2
12 #[1]    1360             3
13 #[1]    364             4
14 #[1]    307             5
15 #[1]      4             6
16 #[1]      2             7
17 #[1]    462             8
18 #[1]    143             9
19 #[1]      2            10
20
21 #_____
22 #for i in range(10) :
23 #     print(pd.DataFrame(commSum2_subg_to[i]).shape)
24
25
26 #number of genes      #community
27 #(5, 1360)            3
28 #(5, 741)             1

```

```

29 #(5, 462)           8
30 #(5, 364)           4
31 #(5, 307)           5
32 #(5, 143)           9
33 #(5, 32)
34 #(5, 28)
35 #(5, 12)
36 #(5, 10)
37
38 #_____
39 #ensLookup_tmpr = list()
40 #for (i in 1:10) {
41 #ensLookup_tmpr[[i]] <- unlist(con_fin1t[i])
42 #print(length(ensLookup_tmpr[[i]]))
43 #}
44
45 #keep genes
46 #from each community:  all      top 30%  community  Table column
47 #_____
48 #[1]      1360      955      3          1
49 #[1]      741      519      1          2
50 #[1]      462      324      8          3
51 #[1]      364      242      4          4
52 #[1]      307      216      5          5
53 #[1]      143      92       9          6
54 #[1]      32       19       7          7
55 #[1]      28       13       8          8
56 #[1]      12       4        9          9
57 #[1]      10       4
58

```