**Generate consensus MERLIN networks and analyze modules**

**Scripts, programs, and files used:**

|  |
| --- |
|  |
| #executable for making consensus network  estEdge=programs/estimateedgeconf/estimateEdgeConf  #executable for making co-clustering matrix  assess=programs/programs/cluster\_analysis/assessclusterstab/assessClusterStab  #executable for hierarchical clustering  reorder=programs/optimalleaforder\_v4/reorder  #executable for zero mean  programs/zeromean.m  # executable for module reordering  module\_reorder=programs/reorderModuleGenes.py  #executable for enrichment:  enrich\_analyzer=programs/enrichanalyzer\_Nongraph\_Qval/enrichAnalyzer  #GO enrichment terms  Pgo=post\_processing/paper\_downloaded\_allgenes\_gop\_regnet.txt  #executable for genClusterAttrib  gen=programs/genClusterAttribConf\_0.5\_merlin\_reg/genClusterAttrib  indir=${location\_of\_MERLIN\_networks} |

**Make consensus network:**

**1. Make a list of network files**

mkdir lists #make sure output directory exist

ls ${indir}/run\*/fold0/prediction\_k100.txt > lists/network\_files.txt

**2. Make consensus network**

- The third option (net\_) is the prefix for output files.

- The program makes two output files, net\_dist.txt, and net\_alledge.txt.

- net\_alledge.txt is the consensus network. It will look like this:

1110038B12Rik    1500012F01Rik    0.99

1110038B12Rik    1500015O10Rik    0.01

1110038B12Rik    2010111I01Rik    0.07

1110038B12Rik    2410006H16Rik    0.81

1110038B12Rik    2810004N23Rik    0.02

0.99 means that edge was observed in 99 out 100 input networks.

mkdir nets

${estEdge} lists/network\_files.txt 0 nets/net\_ alledges

**3. Select edges at different confidence thresholds**

- Here this is showing confidence thresholds of 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, and 0.8.

for i in 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8;

do

cat nets/net\_alledge.txt |awk -v t=$i '$3>=t' > nets/net\_${i}.txt;

done

**4. Format consensus network (target first, TF second) for downstream step for enrichAnalyzer**  
- Note that enrichAnalyzer expect specific keywords in the file name (e.g. \_regnet).

for i in 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8;

do

cat nets/net\_${i}.txt | awk '{printf("%s\t%s\n",$2,$1)}' > nets/merlin\_${i}\_regnet.txt;

done

5. Get run stats:

To select the network to use for visualization, generally we want to select a confidence cutoff where we can retain the highest confidence cutoff while retaining most of the nodes (genes).

for j in 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9;

do

DIR=nets/ # this should be the directory the confidence cutoff networks are in

echo sample ${j}

cd $DIR

python get\_cell\_count.py net\_${j}.txt # count number of nodes in the network

wc -l net\_${j}.txt # count number of edges in the network

echo

done

Abbreviated example output below, where the second line is the number of nodes (genes) and the third line is the number of edges in the network:

sample 0.1

The number of nodes are 13709

57472 net\_0.1.txt

sample 0.2

The number of nodes are 10565

19305 net\_0.2.txt

Here is an example table I generated for confidence cutoffs from the output from above:

|  |  |  |
| --- | --- | --- |
| Confidence  cutoff | Number of nodes | Number of edges |
| All edges | 14216 | 2,920,773 |
| 0.1 | 13709 | 57472 |
| 0.2 | 10565 | 19305 |
| 0.3 | 7502 | 9594 |
| 0.4 | 5076 | 5443 |
| 0.5 | 3385 | 3339 |
| 0.6 | 2257 | 2109 |
| 0.7 | 1480 | 1312 |
| 0.8 | 901 | 746 |
| 0.9 | 456 | 348 |

You could also generate a histogram for confidence cutoffs to visualize nodes per network.

**Make consensus modules:**

**1. Make a list of modules**

ls ${indir}/run\_randpart\*/randpart\*/fold0/modules.txt > module\_files.txt

**2. Make co-clustering matrix**

- element (i,j) of the resulting matrix shows how many times gene i and gene j were in the same cluster

${assess} lists/module\_files.txt lists/sims.txt

**3. Make consensus modules by applying hierarchical clustering to co-clustering matrix**  
- This applies hierarchical clustering to co-clustering matrix

mkdir modules #make sure the output directory exist

for t in 0.1 0.2 0.3 0.4 0.5 0.6

do

${reorder} sims.txt matrix modules/module.$t $t

#you can run this in screen (e.g. "screen -m -d ${reorder} sims.txt matrix modules/module.$t $t"), "-m -d" is to detach the screen automatically after starting the run.

done

**Check number of modules/genes that were returned:**

- This isn’t necessary for the results, but doing this will inform you of the number of genes, number of modules, number of modules with 5 or more genes, and number of genes in those modules.

for i in 0.1 0.2 0.3 0.4 0.5 0.6;   
do   
 a=`cat modules/module.${i}\_geneset.txt |wc -l`; #number of lines in the module file, e.g. number of genes

b=`cat modules/module.${i}\_geneset.txt |cut -f2 |sort -u |wc -l`; #number of unique elements in second column, e.g. number of modules  
#first sort the second column, then count how many of each we have,   
#then filter ones with 5 or more, e.g. number of clusters with 5 or more genes

c=`cat modules/module.${i}\_geneset.txt |cut -f2 |sort |uniq -c |awk '$1>=5' |wc -l`;   
#sum the number of genes in these clusters

d=`cat modules/module.${i}\_geneset.txt |cut -f2 |sort |uniq -c |awk '$1>=5' |awk 'BEGIN{s=0}{s=s+$1}END{printf("%d\n",s)}'`;

echo $i $a $b $c $d;   
done

The output gives the following for each column:

1.  Number of genes

2.  Number of unique elements in second column, e.g. number of modules

3.  Number of clusters with 5 or more genes

4.  Sum the number of genes in these clusters

Here is some example output:

threshold / number of genes / number of modules / number of clusters with 5 or more genes /  sum of genes in clusters

0.1: 14216 5236 258 8257

0.2: 14216 7799 205 5741

0.3: 14216 9535 153 4082

**Do enrichment**  
  
**1. Make genesets (input for enrichAnalyzer)**  
- The output will be in the following format: "Cluster#ID[\t]gene1#gene2#gene3[\n]"

mkdir genesets #in the modules dir, make sure output directory exist

for i in 0.1 0.2 0.3 0.4 0.5 0.6;

do

python2.7 makeGroup.py module.${i}\_geneset.txt genesets/module.${i}.txt 5;

done;

**2. Generate gene list for enrichment (input for enrichAnalyzer)**

We need a background gene list. This example shows using all the genes in modules:

cut -f1 module.0.\*\_geneset.txt |sort -u > bg.txt

**3. Run enrichment**

mkdir enrichments/ # in the modules dir, make sure output directory exist

for t in 0.1 0.2 0.3 0.4 0.5 0.6; #for all the consensus module thresholds

do

#for GO enrichment

${enrich\_analyzer} genesets/module.${t}.txt bg.txt ${Pgo} 0.05 enrichments/go.${t} persg >/dev/null;

for i in 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8; #for all the networks

do

#for expression regulators

${enrich\_analyzer} genesets/module.${t}.txt bg.txt nets/merlin\_${i}\_regnet.txt 0.05 enrichments/merlin.${i}.${t} persg >/dev/null;

done

done

**4. Apply optimal leaf ordering to expression matrix**

- Run reorder.m in matlab to reorder the expression matrix using optimal leaf ordering.

- Then put the header in a separate file

- Change ‘path\_to\_reorder\_script’ to the directory location where the reorder.m script is located.

|  |
| --- |
|  |
| matlab -nosplash -nodesktop -nodisplay -noopengl -r "addpath('${path\_to\_reorder\_script'); reorder('merged3mats\_hvg2folds.txt'); exit"  -The output file is ‘merged3mats\_hvg2folds\_reordered.txt’.  **5. Zero mean the genes in expression matrix.** - Change ‘path\_to\_zeromean\_script’ to the directory location where the zeromean.m script is located. | |

matlab -nosplash -nodesktop -nodisplay -noopengl -r "addpath('${path\_to\_zeromean\_script'); zeromean(merged3mats\_hvg2folds\_reordered.txt'); exit"

# Get header file for step8

head -n1 exp\_reordered\_zeromean.txt |cut -f2- |sed 's/\t/\n/g' > header.txt

#expression without header in a separate file for step 6

cat exp\_reordered\_zeromean.txt > exp\_reordered\_zeromean\_noheader.txt

Output files:

exp\_reordered\_zeromean.txt

header.txt

exp\_reordered\_zeromean\_noheader.txt

6. Reorder modules

- Replace ‘location/of/modules’ at the beginning to the following with the directory where your module files are located.

MODULE\_DIR=location/of/modules

for t in 0.1 0.2 0.3 0.4 0.5 0.6 #for all consensus modules

do

MODULE\_DIR=modules

echo Starting ${t}

echo

python2.7 ${module\_reorder} ${MODULE\_DIR}/enrichments/exp\_reordered\_zeromean\_noheader.txt ${MODULE\_DIR}/${t}\_module\_geneset.txt ${MODULE\_DIR}/${t}\_module\_geneset\_reorder\_zeromean.txt

done

**7. Create input files for downstream steps**  
- We need list of modules and terms enriched in them  
- The following usage shows the -C option, so it is necessary to also make a config file. If you pass them directly to the program, you can instead use -g for GO enrichment and -r expression regulators.

mkdir list\_of\_terms #make output directories

mkdir config

for t in 0.1 0.2 0.3 0.4 0.5 0.6; #for all consensus modules

do

for i in 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8; #for all networks

do

#enriched term (GO or TF) first, cluster ID second, index third (1 for GO, 2 for expression regulators)

#replaces space with "\_" for GO terms

cut -f1,2 enrichments/go.${t}\_details.txt |sed 's/\ /\_/g' |awk '{printf("%s\t%s\t%s\n",$2,$1,1)}' > list\_of\_terms/list.${i}.${t}.txt;

cut -f1,2 enrichments/merlin.${i}.${t}\_details.txt |sed 's/\ /\_/g' |awk '{printf("%s\t%s\t%s\n",$2,$1,2)}' >> list\_of\_terms/list.${i}.${t}.txt;

#location of enrichment files (1 for GO, 2 for expression regulators)

echo enrichments/go.${t}\_details.txt enrichments/merlin.${i}.${t}\_details.txt | awk '{printf("1\t%s\n2\t%s\n",$1,$2)}' > config/conf.${i}.${t}.txt;

done

done

**8. Run genClusterAttrib**

for t in 0.1 0.2 0.3 0.4 0.5 0.6

do

for i in 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8;

do

mkdir -p heatmaps\_in/heat.in.$i.$t #make the output directory

${gen} \

-l list\_of\_terms/list.${i}.${t}.txt \

-o heatmaps\_in/heat.in.$i.$t \

-h header.txt \

-e exp\_reordered\_zeromean\_noheader.txt \

-m modules/module.${t}\_geneset.txt \

-C config/conf.$i.$t.txt

done

done

**9. Make final figures**

- Make sure Heatmap.awk exists in the directory you execute the script from  
- Execute the following in the directory that the ‘heat.in.\*’ directories are located at.

for t in 0.1 0.2 0.3 0.4 0.5 0.6 #for all consensus modules

do

for f in 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 #for all networks

do

echo Starting ${t}

bash makeHeatmap\_colormap.sh heat.in.${f}.${t}

echo

echo

done

done

Output heatmaps will be in the ‘heatmaps\_out’ directory.

**Get Fscore and recall heatmaps**

* This script will generate 2 heatmaps for fscore and recall.
* You will have to change file name prefixes on lines 8, 10, 24, and 26 (file name except for ‘.txt’) depending on the file names of the networks and gold standards you are comparing. You may also have to change the file path as well for the networks and gold standards.
* You might have to change the file path to the ‘validate’ file path on lines 17 and 32 if not running the program from the ‘post\_processing’ dir.
* You might have to change the file path to the ‘Heatmap.awk’ file path on lines 21 and 36 if not running the program from the ‘post\_processing’ dir.

bash makeSvg\_recall\_fscore\_merlin.sh

Two heatmap files will be generated:

* recall.svg
* fscore.svg