Mergeomics: Integrative Network Analysis of Omics Data

April 27, 2020

Ville-Petteri Makinen, Le Shu, Yuqi Zhao, Zeyneb Kurt, Bin Zhang, Xia Yang
Department of Integrative Biology and Physiology, University of California, Los Angeles
South Australian Health and Medical Research Institute
Department of Genetics and Genomics Sciences, Mount Sinai School of Medicine

zhaoyuqi616@ucla.edu, zeyneb@ucla.edu, icestrike@ucla.edu, xyang123@ucla.edu

If you use mergeomics in published research, please cite:
Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

Contents

1 Introduction

The recent revolution in genomic technologies has enabled the generation of massive amount of molecular data encompassing genetic, transcriptomic, epigenomic, metabolomics, and proteomics, which have become easily accessible in public domains and private sectors. It is increasingly recognized that analysis of individual types of data separately only reveals a fraction of the complex biology and often misses the key players driving diseases, making multi-dimensional big data integration an urgent need. Mergeomics package is developed as a capable and flexible pipeline to integrate various types of disease association data such as genetic association (e.g, GWAS or exome sequencing), transcriptome-wide association (e.g., TWAS from microarray or RNA sequencing studies), and epigenetic association (e.g., EWAS from methylome association studies), functional genomics (such as eQTLs and ENCODE annotations), biological pathways, and gene networks to identify disease-associated gene sub-networks and key regulatory genes.

In this tutorial, we will assume that you have downloaded the standalone Mergeomics package from

http://mergeomics.research.idre.ucla.edu/Download/Package/

Mergeomics.tar.gz and install it using the following commands:

```
> install.packages("Mergeomics.tar.gz", repos = NULL,
+ type="source")
> ## or from bioconductor3.3 release, use following lines:
> if (!requireNamespace("BiocManager", quietly=TRUE))
+     install.packages("BiocManager")
> BiocManager::install("Mergeomics")
```

However, if you want, you can also try out these examples with our webserver at http://mergeomics.research.idre.ucla.edu. If you have any questions regarding this document or the described software, please contact us: Xia Yang (xyang123@ucla.edu).

2 Tutorials

The whole Mergeomics pipeline can be divided into two major components: 1) Marker Set Enrichment Analysis (MSEA), 2) Weighted Key Driver Analysis (wKDA). In the following step, we will illustrate how to implement Mergeomics using one sample dataset, the descriptions of which are as follows.

2.1 Input Data Preparation

In this part we described the datasets used by Mergeomics. Mergeomics is designed to integrate summary level data or curated resources, thus light data-preprocessing work is expected for user to convert datasets into Mergeomics ready format. For the convenience of users, we also deposited a rich collection of datasets that can be directly used for analysis in the Mergeomics webserver. Of note, Mergeomics is not able to directly process raw data from omics studies (e.g., sequencing reads).

Mergeomics calls for five types of datasets. All files should contain tab delimited plain text.

1. Marker-disease association summary result: Summary result files from association studies such as GWAS, TWAS or EWAS. Only two columns are required, MARKER and VALUE. The MARKER column gives the IDs of the markers (e.g., reference SNP ID), whereas the VALUE column contains the -log10 transformed association P values for the markers. The package provides a sample summary file from the Genome-wide association studies (GWAS) of high-density lipoprotein cholesterol (HDLC) from Global Lipids Genetics Consortium [1].

2. Gene-marker mapping file: Data that defines the mapping of genes and markers. Two columns are required, GENE and MARKER. Common resources for the file is chromosomal distance based mapping, eQTL based mapping and ENCODE based mapping. The Mergeomics webserver provides a comprehensive collection of above-mentioned mapping files. Experienced users can also generate their own mapping files from publicly available resources such as GTEx and ENCODE. The package provides a sample mapping file where the GWAS loci were projected to genes within 40 Kb distance from gene region.

3. Functionally related gene-sets: Two columns are required, MODULE and GENE. The MODULE column defines the grouping of genes, normally by their functional relatedness. Gene-sets can be derived from canonical pathways, co-expression modules or differential genes under experimental

conditions. The sample gene-sets in the package is the coexpression networks re-constructed in mouse/human liver tissues, containing 2674 modules[2-6].

4. Gene regulatory networks: Three columns are required, HEAD, TAIL and WEIGHT. If the user is using a directional network, then the starting nodes should be put in the HEAD column, while the ending nodes should be put in the TAIL column. If a non-directional network is used, users have to set the corresponding wKDA parameter to "non-directional", and are then safe to switch the HEAD and TAIL column. The WEIGHT column normally contains the confidence score or consensus score (if users merge multiple networks into one) of edges. If no weight information is available, simply fix all WEIGHT value to 1. The packages provides a sample Bayesian networks generated from mouse/human liver tissues [2-6].

5. Marker dependency structure file (Optional): This file is needed if users plan to use the MDPrune module. Three columns are required, MARKERa, MARKERb and WEIGHT. The WEIGHT column describes the degree of dependency (e.g., linkage disequilibrium R2) between MARKERa and MARKERb.

2.2 MDPrune

Before taking any further steps, we recommend users to preprocess datasets to filter dependency structure among markers (e.g. Linkage disequilibrium), as marker dependency may cause artifacts and biases in the downstream analysis. We have provided an external C++ program named MDPrune (http://mergeomics.research.idre.ucla.edu/Download/MDPrune/), which can remove marker dependency while preferentially keeping those with a strong statistical association with traits.

The commands are listed as follows:

mdprune MARFile MAPFile MDSFile OutPath [Cutoff]

The file format request for MDPrune is the same as main Mergeomics pipeline, where;

MARFile is the association study summary file

MAPFile is the Gene-marker mapping file;

MDSFile is the marker dependency structure file.

Here, we chose a moderate cutoff (R^2>0.7) for the dataset.

2.3 One-step Mergeomics

In this section, we first provide an one-step way to run the whole pipeline. One-step running of the pipeline can be accomplished either with a built-in function including all function calls in it or by explicitly calling/running all the required functions related with the pipeline processes one-by-one sequentially.

We demonstrated the either ways of one-step Mergeomics pipeline with the code chunks given in this section. Then, the step-by-step processing of the pipeline will be described in detail in separate sections.

The first way of one-step pipeline calling is given in the following code chunk. It is achieved by one built-in function call (MSEA.KDA.onestep).

```
> ############################################################
> #######    One-step analysis for Mergeomics - 1st way    ##
> ############################################################
> ## Import library scripts.
> library(Mergeomics)
> ## first, give the module info file, genefile, marker file, and network file
> ## paths for the pipeline
> plan <- list()
> plan$label <- "hdlc"
> plan$folder <- "Results"
> plan$genfile <- system.file("extdata",
+ "genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
> plan$marfile <- system.file("extdata",
+ "marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
> plan$modfile <- system.file("extdata",
+ "modules.mousecoexpr.liver.human.txt", package="Mergeomics")
> plan$inffile <- system.file("extdata",
+ "coexpr.info.txt", package="Mergeomics")
> plan$netfile <- system.file("extdata",
+ "network.mouseliver.mouse.txt", package="Mergeomics")
> ## second, define pipeline parameters (e.g. permutation #, seed for random #
> ## generator, min and max module sizes, max overlapping ratio, etc.)
> plan$permtype <- "gene" ## default setting is gene permutation
> plan$nperm <- 100 ## default value is 20000
> plan$mingenes <- 10   ## default value is 10
> plan$maxgenes <- 500   ## default value is 500
> ## then, call the one-step pipeline function including both MSEA and KDA steps
> plan <- MSEA.KDA.onestep(plan, apply.MSEA=TRUE, apply.KDA=TRUE,
```

```
+ maxoverlap.genesets=0.20, symbol.transfer.needed=TRUE,
+ sym.from=c("HUMAN", "MOUSE"), sym.to=c("HUMAN", "MOUSE"))
> ## NOTE: default value of maxoverlap.genesets=0.33; but in all the examples of
> ## this tutorial it is 0.2 (see job.msea$rmax<- 0.2 in the following example)
> ## maxoverlap.genesets=0.33 (or job.msea$rmax<- 0.33) is recommended to use.
```

The second way of one-step pipeline calling is given in the following code chunk. Here, we need to explicitly and sequentially call main functions of both MSEA and KDA processes.

```
> ############################################################
> #######    One-step analysis for Mergeomics - 2nd way    ##
> ############################################################
> ## Import library scripts.
> library(Mergeomics)
> ################ MSEA (Marker set enrichment analysis)  ###
> job.msea <- list()
> job.msea$label <- "hdlc"
> job.msea$folder <- "Results"
> job.msea$genfile <- system.file("extdata",
+ "genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
> job.msea$marfile <- system.file("extdata",
+ "marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
> job.msea$modfile <- system.file("extdata",
+ "modules.mousecoexpr.liver.human.txt", package="Mergeomics")
> job.msea$inffile <- system.file("extdata", "coexpr.info.txt",
+ package="Mergeomics")
> job.msea$nperm <- 100 ## default value is 20000 (this is recommended)
> job.msea <- ssea.start(job.msea)
> job.msea <- ssea.prepare(job.msea)
> job.msea <- ssea.control(job.msea)
> job.msea <- ssea.analyze(job.msea)
> job.msea <- ssea.finish(job.msea)
> #########  Create intermediary datasets for KDA ###########
> syms <- tool.read(system.file("extdata", "symbols.txt",
+ package="Mergeomics"))
> syms <- syms[,c("HUMAN", "MOUSE")]
> names(syms) <- c("FROM", "TO")
> ## default and recommended rmax=0.33.
> ## min.module.count is the number of the pathways to be taken from the MSEA
```

```
> ## results to merge. If it is not specified (NULL), all the pathways having
> ## MSEA-FDR value less than 0.25 will be considered for merging if they are
> ## overlapping with the given ratio rmax.
> job.kda <- ssea2kda(job.msea, rmax=0.2, symbols=syms, min.module.count=NULL)
> #######   wKDA (Weighted key driver analysis)    ##########
> job.kda$netfile <- system.file("extdata",
+ "network.mouseliver.mouse.txt", package="Mergeomics")
> job.kda <- kda.configure(job.kda)
> job.kda <- kda.start(job.kda)
> job.kda <- kda.prepare(job.kda)
> job.kda <- kda.analyze(job.kda)
> job.kda <- kda.finish(job.kda)
> ######  Prepare network files for visualization   #########
> ## Creates the input files for Cytoscape (http://www.cytoscape.org/)
> job.kda <- kda2cytoscape(job.kda)
```

After that, we focused on each pipeline process in the following sections.

2.4 Marker set enrichment analysis (MSEA)

The purpose of the MSEA is to leverage genome/epigenome wide association data, function genomics, canonical pathways and/or data-driven gene modules for identifying causal subnetworks for disease/traits.

```
> ############################################################
> ## Import Mergeomics library.
> library("Mergeomics")
> ## create an empty list for setting parameters
> job.msea <- list()
> ## Next, label your project
> job.msea$label <- "HDLC"
> ## The pathway size varies from 1 to a few thousands and will
> ## introduce bias to the analysis. We set criteria for the
> ## min. (mingenes) and max. (maxgenes) gene size for the pathways.
> job.msea$maxgenes <- 500
> job.msea$mingenes <- 10
> ## The permutation setting and number of permutations. We recommend
> ## using gene permutation due to its robust performance.
> ## The alternative is locus permutation.
> job.msea$permtype <- "gene"
```

```
> job.msea$nperm <- 100 ## default value is 20000 (this is recommended)
> ## set the output folder
> job.msea$folder <- "./Results"
> ## The parameter genfile defines the Marker-to-Gene mapping file
> ## It contains two columns, GENE and MARKER, delimited by tab
> job.msea$genfile <- system.file("extdata",
+ "genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
> ## The parameter marfile defines the Disease association data file
> ## It contains two columns, MARKER and VALUE, delimited by tab
> ## Here, the marfile comes from the GWAS file after marker
> ## dependency pruning, so the VALUE is the minus log10 transformed
> job.msea$marfile <- system.file("extdata",
+ "marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
> ## The modfile defines the pathway information, which could come
> ## from knowledge-based databases (such as KEGG, and Reactome)
> ## or data-driven data sets (such as co-expression modules).
> ## It contains two columns, MOUDLE and GENE, delimited by tab
> job.msea$modfile <- system.file("extdata",
+ "modules.mousecoexpr.liver.human.txt", package="Mergeomics")
> ## The inffile provides the basic descriptions for the pathways
> ## It contains three columns, MODULE, SOURCE, and DESCR, which
> ## provide information for pathway IDs corresponding to the
> ## pathway names in modfile, the sources of the pathways, and
> ## pathway annotations
> job.msea$inffile <- system.file("extdata", "coexpr.info.txt",
+ package="Mergeomics")
> ## Then, MSEA will run for ~30 minutes to ~2 hours
> job.msea <- ssea.start(job.msea)
> job.msea <- ssea.prepare(job.msea)
> job.msea <- ssea.control(job.msea)
> job.msea <- ssea.analyze(job.msea)
> job.msea <- ssea.finish(job.msea)
> ############################################################
```

A part of the original results is listed in Table 1. The pathways satisfying certain cutoffs (e.g. FDR<30%) will be used for the next step and can be further annotated using diverse tools, such as DAVID (http://david.abcc.ncifcrf.gov/).

Table 1: Summary result of MSEA

Pathways	Pvalues	FDRs	Descriptions
5099	3.48E-04	6.05%	5099:liver
4588	1.05E-03	13.64%	4588:liver
4737	2.01E-03	21.02%	4737:liver
4128	4.44E-03	30.24%	4128:liver
5117	5.22E-03	32.03%	5117:liver
6645	5.24E-03	32.05%	6645:liver
5104	5.50E-03	32.29%	5104:liver
4094	7.65E-03	33.93%	4094:liver
4932	7.95E-03	34.12%	4932:liver

2.4.1 ModuleMerge

Usually, pathways/modules collected from different sources will have certain degree of redundancies, or refer to the same processes. In this case, users could merge the disease-related pathways into relatively independent gene sets, which exceed a given FDR cutoff after running MSEA.

```
> ############################################################
> job <-list()
> job$folder <- c("module_merge")
> ## The moddata and modinfo either come from the significant pathways found in
> ## any previous MSEA run or these files can be manually curated by the user.
> ## moddata is an obligatory input file; while modinfo is an optional input.
> ## It is recommended to merge the overlapping pathways among significant
> ## ones before applying KDA to proceed with the independent gene sets.
> moddata <- tool.read(system.file("extdata", "Significant_pathways.txt",
+ package="Mergeomics"), c("MODULE","GENE"))
> modinfo <- tool.read(system.file("extdata", "Significant_pathways.info.txt",
+ package="Mergeomics"),c("MODULE","SOURCE","DESCR"))
> syms <- tool.read(system.file("extdata", "symbols.txt",
+ package="Mergeomics"))
> syms <- syms[,c("HUMAN", "MOUSE")]
> names(syms) <- c("FROM", "TO")
> moddata$GENE <- tool.translate(words=moddata$GENE, from=syms$FROM,
+ to=syms$TO)
> ## Merge and trim overlapping modules.
> rmax <- 0.2
```

```
> moddata$OVERLAP <- moddata$MODULE
> moddata <- tool.coalesce(items=moddata$GENE, groups=moddata$MODULE,
+ rcutoff=rmax)
> moddata$MODULE <- moddata$CLUSTER
> moddata$GENE <- moddata$ITEM
> moddata$OVERLAP <- moddata$GROUPS
> moddata <- moddata[,c("MODULE", "GENE", "OVERLAP")]
> moddata <- unique(moddata)
> modinfo <- modinfo[which(!is.na(match(modinfo[,1], moddata[,1]))), ]
> ## Mark modules with overlaps.
> for(i in which(moddata$MODULE != moddata$OVERLAP)){
+     modinfo[which(modinfo[,"MODULE"] == moddata[i,"MODULE"]),
+             "MODULE"] <- paste(moddata[i,"MODULE"], "..", sep=",")
+     moddata[i,"MODULE"] <- paste(moddata[i,"MODULE"], "..", sep=",")
+ }
> ## Save merged module data and info for KDA.
> modfile <- "mergedModules.txt"
> modinfofile <- "mergedModules.info.txt"
> moddata$NODE <- moddata$GENE
> tool.save(frame=unique(moddata[,c("MODULE", "GENE", "OVERLAP", "NODE")]),
+ file=modfile, directory=job$folder)
> tool.save(frame=unique(modinfo[,c("MODULE","SOURCE","DESCR")]),
+ file=modinfofile, directory=job$folder)
> ############################################################
```

Users are also recommended to analyze the merged genesets with MSEA again to confirm that they are still significantly associated with disease.

2.4.2 Meta-MSEA

When multiple disease association datasets are available, meta-MSEA can be used to conduct meta-analysis at pathway or network level. This function allows user to achieve maximal power by combining results from independent association studies of different ethnicity, platform or even species, while evading the technical difficulties when performing meta-analysis directly on the marker-level association data.

```
> ############################################################
> ## Assume there are three MSEA objects passed down by
> ## ssea.finish()
> # job.metamsea = list()
```

```
> # job.metamsea$job1 = job.msea1
> # job.metamsea$job2 = job.msea2
> # job.metamsea$job3 = job.msea3
> # job.metamsea = ssea.meta(job.metamsea,"meta_label",
> # "meta_folder")
> ############################################################
```

2.5 Weighted key driver analysis (wKDA)

wKDA aims to pinpoint key regulator genes (or key drivers) of the disease related gene sets using gene network topology and edge weight information. In specific, wKDA first screen the network for candidate hub genes. Then the disease gene-sets are overlaid onto the subnetworks of the candidate hubs to identify key drivers whose neighbors are enriched with disease genes.

```
> ############################################################
> job.kda <- list()
> job.kda$label<-"HDLC"
> ## parent folder for results
> job.kda$folder<-"./Results"
> ## Input a network
> ## columns: TAIL HEAD WEIGHT
> job.kda$netfile <- system.file("extdata", "network.mouseliver.mouse.txt",
+ package="Mergeomics")
> ## Gene sets derived from ModuleMerge, containing two columns,
> ## MODULE, NODE, delimited by tab
> job.kda$modfile<- system.file("extdata",
+ "mergedModules.txt", package="Mergeomics")
> ## Annotation file for the gene sets
> ## if module or pathway annotation is not available, skip this:
> job.kda$inffile<-system.file("extdata",
+ "mergedModules.info.txt", package="Mergeomics")
> ## "0" means we do not consider edge weights while 1 is
> ## opposite.
> job.kda$edgefactor<-0.5 ## default value
> ## The searching depth for the KDA
> job.kda$depth<-1 ## default value
> ## "0" means we do not consider the directions of the
> ## regulatory interactions
> ## while 1 is opposite.
```

Table 2: Summary result of wKDA

Key Drivers	MODULES	Pvalues	FDRs	Descriptions
Itih4	4932,..	2.90E-14	0.00%	4932:liver
Clstn3	136	6.92E-14	0.00%	136:liver
Pmvk	4407	1.40E-13	0.00%	4407:liver
Acss2	4407	1.54E-13	0.00%	4407:liver
Gpam	4407	3.82E-13	0.00%	4407:liver
Aqp8	4407	8.26E-13	0.00%	4407:liver
Itih4	5099	1.56E-12	0.00%	5099:liver
Lpin1	4084	2.21E-12	0.00%	4084:liver

```
> job.kda$direction<-0 ## default value
> ## Let us run KDA!
> job.kda <- kda.configure(job.kda)
> job.kda <- kda.start(job.kda)
> job.kda <- kda.prepare(job.kda)
> job.kda <- kda.analyze(job.kda)
> job.kda <- kda.finish(job.kda)
> ############################################################
```

2.6 Network visualization

For the current version, we generate Cytoscape-ready (http://www.cytoscape.org/) input files to visualize the network of key driver (KD) genes. Cytoscape-ready files are generated for illustrating either the neighborhoods of top five KDs of each module or the neighborhoods of a particular gene (node) list. By default, log files for each module's top five KDs' neighborhoods are prepared. Number of the top KDs can be specified by the user. Node and edge list files are created separately for Cytoscape. Additionally, top key driver name list for each module and color map of the modules are logged in the relevant files. kda2cytoscape function provides users:

- To illustrate top five KD neighborhoods for the modules, whose pathway or module name is listed within "modules" parameter (all modules are selected by default),
- To specify a particular number of KDs for each module with "ndrivers" (default value is five),

- To search and illustrate the neighborhoods of a given node list with "node.list" parameter (default value is NULL; if this is not specified by the user, top 5 KDs of each module and their neighborhoods will be illustrated; if "node.list" parameter is specified, it will be prioritized rather than top 5 KDs illustration),

- To examine the KD neighborhoods for a given depth (default depth is 1; hence, only the directly connected neighbor nodes of key drivers are taken into account).

```
> ############################################################
> ## run following line after finishing the KDA process
> ## i.e., after the kda.finish() function concluded
> job.kda <- kda2cytoscape (job.kda, node.list=NULL,
+ modules=NULL, ndrivers=5, depth=1)
> ############################################################
```

Generated file names and their descriptions are given below:

- **kda2cytoscape.top.kds.txt:** Top KDs of the modules are listed in this file. Number of the key drivers can be set by user with "ndrivers" parameter.

- **kda2cytoscape.edges.txt:** Edge lists of the generated graph. This file should be imported as network within the Cytoscape.

- **kda2cytoscape.nodes.txt:** Node lists of the generated graph. This file includes the optional formatting information for the nodes of the graph. It should be imported as node table within the Cytoscape. Field/attribute list provided for each node is as follows:

  - NODE: gene symbol
  - LABEL: node label (same as above)
  - COLOR: background color of the node, it also shows the module membership of the node. If a node is member of multiple modules/pathways, only one color among them will be assigned as background color, but in the URL field this multiple module membership will be represented by a pie chart (node is divided into several sectors to show the multi-membership of the node). If a node is not a member for any of the pathways, the color will be "grey"

  - SIZE: size of the node. If the node is a KD it will be twice larger than ordinary nodes.
  - SHAPE: shape of the node. If the node is a KD, it will be in diamond-shape, otherwise it is illustrated with circle.
  - URL: pie chart created by Google chart tools. If the node is member of multiple modules, node circle will be divided as a pie chart and colored by the module colors it belongs to
  - LABELSIZE: text size of the node. If the node is a KD, its font size will be larger than ordinary nodes.

- **module.color.mapping.txt:** Color mapping for the modules, i.e. one color is assigned to each module.

An example graph for top 5 KDs'neighborhoods of a given pathway is illustrated in Figure 1. Different colors represent modules/pathways, while the larger nodes are the key driver genes.

Note that: SHAPE feature is not used in the Cytoscape in Fig 1. All the listed attributes for nodes are optional.

3 Citation

If you use Mergeomics in published work, please cite: Shu L, Zhao Y, Kurt Z, Byars S, Tukiainen T, Kettunen J, Ripatti S, Zhang B, Inouye M, Makinen VP, Yang X. Mergeomics: integration of diverse genomics resources to identify pathogenic perturbations to biological systems.

bioRxiv doi: http://dx.doi.org/10.1101/036012.

4 References

[1] Willer CJ, Schmidt EM, Sengupta S, Peloso GM, Gustafsson S, et al. (2013) Discovery and refinement of loci associated with lipid levels. Nature Genetics 45.

[2] Derry JMJ, Zhong H, Molony C, MacNeil D, GuhaThakurta D, et al. (2010) Identification of genes and networks driving cardiovascular and metabolic phenotypes in a mouse F2 intercross. PLoS ONE 5: e14319. doi:10.1371/journal.pone.0014319.

[3] Wang SS, Schadt EE, Wang H, Wang X, Ingram-Drake L, et al. (2007) Identification of pathways for atherosclerosis in mice: integration of

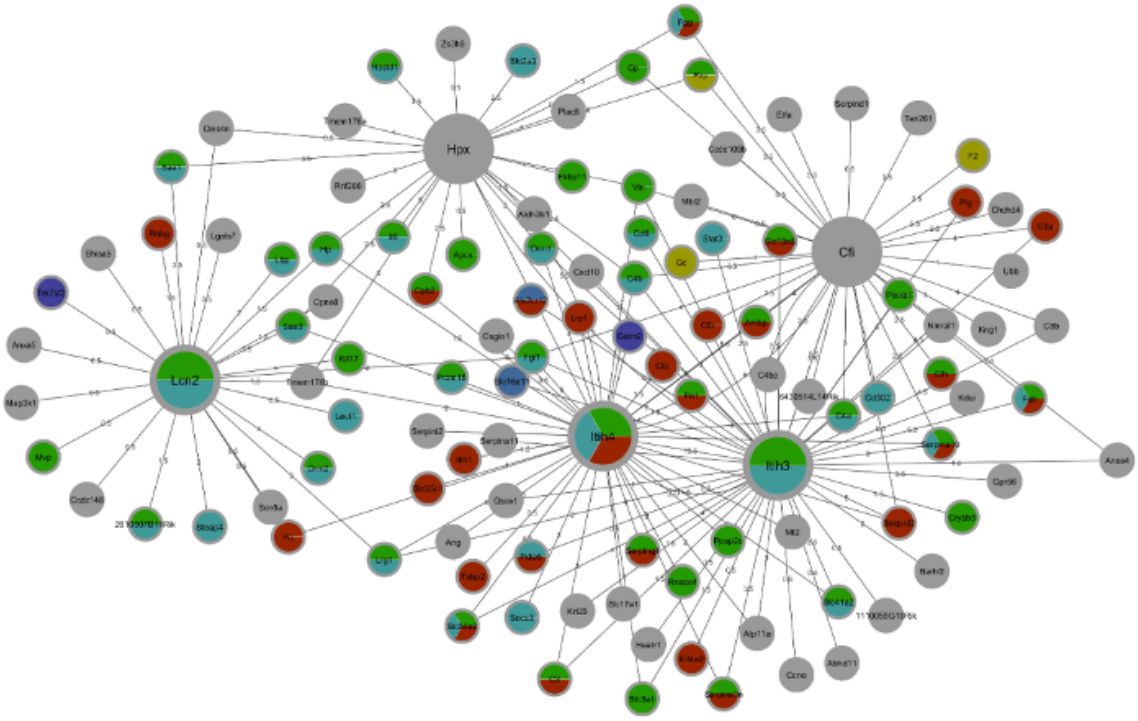

Figure 1: The key driver gene subnetwork

quantitative trait locus analysis and global gene expression data. Circ Res 101: e11-e30. doi:10.1161/CIRCRESAHA.107.152975.

[4] Yang X, Schadt EE, Wang S, Wang H, Arnold AP, et al. (2006) Tissue-specific expression and regulation of sexually dimorphic genes in mice. Genome Res 16: 995-1004. doi:10.1101/gr.5217506.

[5] Schadt EE, Molony C, Chudin E, Hao K, Yang X, et al. (2008) Mapping the genetic architecture of gene expression in human liver. PLoS Biol 6: e107. doi:10.1371/journal.pbio.0060107.

[6] Tu ZZ, Keller MPM, Zhang CC, Rabaglia MEM, Greenawalt DMD, et al. (2012) Integrative analysis of a cross-loci regulation network identifies App as a gene regulating insulin secretion from pancreatic islets. PLoS Genet 8: e1003107-e1003107. doi:10.1371/journal.pgen.1003107.