mmnet: Metagenomic analysis of microbiome metabolic network

Yang Cao, Fei Li, Xiaochen Bo

February 1, 2016

Contents

1 Introduction

This manual is a brief introduction to structure, functions and usage of *mmnet* package. The *mmnet* package provides a set of functions to support systems analysis of metagenomic data in R, including annotation of raw metagenomic sequence data, construction of metabolic network, and differential network analysis based on state specific metabolic network and enzymatic gene abundance.

Meanwhile, the package supports an analysis pipeline for metagenomic systems biology. We can simply start from raw metagenomic sequence data, optionally use MG-RAST to rapidly retrieve metagenomic annotation, fetch pathway data from KEGG using its API to construct metabolic network, and then utilize a metagenomic systems biology computational framework mentioned in (**?**) to establish further differential network analysis.

The main features of *mmnet*:

- Annotation of metagenomic sequence reads with MG-RAST
- Estimating abundances of enzymatic gene based on functional annotation
- Constructing State Specific metabolic Network
- Topological network analysis
- Differential network analysis

1.1 Installation

mmnet requires these packages: *KEGGREST*, *igraph*, *Biobase*, *XML*, *RCurl*, *RJSONIO*, *stringr*, *ggplot2* and *biom*. These should be installed automatically when you install *mmnet* with biocLite() as follows:

```
## install release version of mmnet
source("http://bioconductor.org/biocLite.R")
biocLite("mmnet")

## install the latest development version
```

```
useDevel()
biocLite("mmnet")
```

It should be noted that the function call useDevel() calling is chechout to the development versino of BIOCONDUCTOR and it will be install the latest development version of mmnet, which has more functions available and a lot of bugs fixed.

After installation, the *mmnet* is ready to load into the current workspace by the following codes to the current workspace by typing or pasting the following codes:

```
library(mmnet)
```

2 Analysis Pipeline: from raw Metagenomic Sequence Data to Metabolic Network Analysis

We will demonstrate go through an analysis pipeline to illustrate some of the main functions in *mmnet*. This pipeline consists of several steps:

1. Raw metagenomic sequence data: prepare a specific amount of sequence data ready to analyze from metagenomic studies. You can get the data by short-gun sequencing or download raw data available online. Here we download a sample data from ftp of MG-RAST.
2. Functional annotation: raw reads must be processed for functional annotation to mapping to enzyme libraries with existing knowledge, such as KEGG database.
3. State Specific Network: start from an enzyme set, we can map these enzymes onto metabolic pathways to construct a relatively small metabolic network with abundance involved in specific biological states, named State Specific Network (SSN).
4. Topological and differential network analysis: topological network analysis performs a series of correlation analysis between enzyme abundances and topological properties, i.e. betweenness or pageRank. Meanwhile differential network analysis compares the enzyme abundances and structure of different state specific networks.

2.1 Prepare metagenomic sequence data

Acceptable sequence data can be in FASTA, FASTQ or SFF format. These are recognized by the file name extension with valid extensions for the appropriate formats .fasta, .fna, .fastq, .fq, and .sff and FASTA and FASTQ files need to be in plain text ASCII. An easy way to get some sequence data is to download public dataset from MG-RAST. Here we download two FASTA data sets from MG-RAST ftp server, which corresponds two metagenomic samples with different physiological states (MG-RAST ID: 4440616.3 and 4440823.3). These data sets are consisting of several microbiomes from twin pairs and their mothers (**?**).

```
download.file("ftp://ftp.metagenomics.anl.gov/projects/10/4440616.3/raw/507.fna.gz",
    destfile = "obesesample.fna.gz")
download.file("ftp://ftp.metagenomics.anl.gov/projects/10/4440823.3/raw/687.fna.gz",
    destfile = "leansample.fna.gz")
```

2.2 Annotation of Metagenomic Sequence Reads

Functional annotation of microbiome is a crucial step for subsequent analysis. Many tools can be used to identify functions of sequences. Meanwhile, local analysis tools and annotation database need complicated installation and configuration, and suppose users have deep understanding on high performance computing and next generation sequencing, that requires users have sufficient preparing time, technical kills and computing resource.

To facilitate the analysis of microbiome metabolic network, *mmnet* provides functions for sequence annotation linking *R* with MG-RAST. MG-RAST (**?**) is a stable, extensible, online analysis platform for annotation and statistic metagenomes based on sequence data. Moreover, MG-RAST is freely available to all researchers. For more details, see `http://metagenomics.anl.gov`.

By integrating the web service provided by MG-RAST into R environment. *mmnet* provides several functions for assigning functional annotation to reads : *loginMgrast*,*uploadMgrast*, *submitMgrastJob*, *listMgrastProject* and *checkMgrastMetagenome*.

Before use the MG-RAST services, we need register and login a MG-RAST account to access its service. Registration needs to be done on the website. Login can be processed by function *loginMgrast*. After successful calling this function with user's name and password, a session variable login.info is returned for further access to MG-RAST service. It should be noted that username 'mmnet' with password 'mmnet' was already registered for testing purpose. See '?loginMgrast' for more details.

```
## login on MG-RAST
login.info <- loginMgrast(user = "mmnet", userpwd = "mmnet")
```

After login, the next step is to upload raw sequence data (obesesample.fna.gz and leansample.fna.ga in here) to your MG-RAST inbox. Function *uploadMgrast* is designed to get the job done easily.

```
## select the sample data to upload
seq.file <- c("obesesample.fna.gz", "leansample.fna.gz")
## upload sequence data to MG-RAST
metagenome.id <- lapply(seq.file, uploadMgrast, login.info = login.info)
```

Note: According to MG-RAST user manual, uploaded files may be removed from your inbox after 72 hours. So please perform submission of your files within that time frame.

Once the sequence data is uploaded to your MGRAST inbox, users could submit one MGRAST job to annotate each sequence files in your inbox with function *submitMgrastJob*.

```
## submit MGRAST job
metagenome.id <- lapply(seq.file, submitMgrastJob,
    login.info = login.info)
show(metagenome.id)
```

It should be noticed that our sample metagenomic sequences downloaded from the MG-RAST have been annotated, so it will return the corresponding metagenome ID that already exists in MG-RAST without duplicated annotation.

In other cases, the annotation process may take several hours or days. Thus, here we provide a function *checkMgrastMetagenome* to check whether your metagenome annotation is completed or in processing.

```
## check MGRAST project status
metagenome.status <- lapply(metagenome.id, checkMgrastMetagenome,
    login.info = login.info)
## apparently, status of completed annotation
## metagenome is TRUE
show(metagenome.status)
```

Once the annotation process completes, the metagenome ID can be obtained by function *getMgrastAnnotation* to load the functional annotation profile separately for subsequent metabolic analysis. For private data, you must login on MGRAST to get login.info and call *getMgrastAnnotation* with login.info. For more details see *'getMgrastAnnotation'*.

```
## private data
private.annotation <- lapply(metagenome.id, getMgrastAnnotation,
    login.info = login.info)
## public annotation data, does not require
## login.info
public.annotation <- lapply(metagenome.id, getMgrastAnnotation)
```

For convenience, we have save this annotation profiles in *mmnet* names *'anno.rda'*. See *'anno'* for more details. Loading annotation profiles as follows:

```
data(anno)
summary(anno)

##           Length Class      Mode
## 4440616.3 13     data.frame list
## 4440823.3 13     data.frame list
```

Here, we show an entire process which integrates all the functions above, from sequence uploading to annotation profile downloading, and retrieve the annotation profile after MG-RAST annotation completed.

```
## first login on MG-RAST
login.info <- loginMgrast("mmnet", "mmnet")
## prepare the metagenomic sequence for annotation
seq <- "obesesample.fna.gz"
## mgrast annotation
uploadMgrast(login.info, seq)
metagenome.id2 <- submitMgrastJob(login.info, seqfile = basename(seq))
while (TRUE) {
    status <- checkMgrastMetagenome(metagenome.id = metagenome.id2)
    if (status)
        break
    Sys.sleep(5)
    cat("In annotation, please waiting...")
    flush.console()
}
## if annotation profile is public,take login.info
## as NULL
anno2 <- getMgrastAnnotation(metagenome.id2, login.info = login.info)
```

2.3 Estimating the abundance of enzymatic genes

Enzyme abundance varies in different biological states. Enzymatic gene abundances are important for microbial marker-gene and disease study, especially for linking microbial genes with the host state. Here, we estimate enzymatic gene abundances based on read counts with a normalization procedure.

To estimate the abundance of enzymatic genes, *estimateAbundance* function should be called for each sample. A widely used in metagenomic analysis format - Biological Observation Matrix (BIOM) (**?**) format will be return by this function. The BIOM file format (canonically pronounced biome) is designed to be a general-use format for representing biological sample by observation contingency tables. More details on BIOM file can be found in `http://biom-format.org/`.

```
mmnet.abund <- estimateAbundance(anno)
show(mmnet.abund)

## biom object.
## type: enzymatic genes abundance
## matrix_type: dense
## 1034 rows and 2 columns
```

Furthermore, BIOM abundance file was also supported by other tools. Take MG-RAST for example, user could download the enzymatic genes abundance profile with MG-RAST API. Moreover, while samples have been annotated with other tools (e.g. blast to KEGG locally), users could create their own BIOM files represents enzymatic genes abundance and import them for metagenomic systems biology analysis.

```
## download BIOM functional abundance profile of the
## two sample metagenome from MG-RAST
if (require(RCurl)) {
    function.api <- "http://api.metagenomics.anl.gov/1/matrix/function"
    mgrast.abund <- read_biom(getForm(function.api,
        .params = list(id = "mgm4440616.3", id = "mgm4440823.3",
            source = "KO", result_type = "abundance")))
}
## obtain the intersect ko abundance of MG-RAST and
## esimatiAbundance
intersect.ko <- intersect(rownames(mgrast.abund), rownames(mmnet.abund))
## compare the two by taking one metagenome
mgrast.abund1 <- biom_data(mgrast.abund)[, 1][intersect.ko]
mmnet.abund1 <- biom_data(mmnet.abund)[, 1][intersect.ko]
if (require(ggplot2)) {
    p <- qplot(mgrast.abund1, mmnet.abund1) + geom_abline(slope = 1,
        intercept = 0, color = "blue") + ylim(0, 400) +
        xlim(0, 400)
    print(p)
}
```

The enzymatic gene abundances estimated by MG-RAST is un-calibrated and un-optimized compared to the method in *mmnet*. As show in Figure 1, abundances in MG-RAST is over-estimated in most cases, especially on high abundance enzymatic genes. The details on abundances calibration in *mmnet* is as follows: Sequences were annotated with all KOs (KEGG orthologous groups) of the top KO-associated match

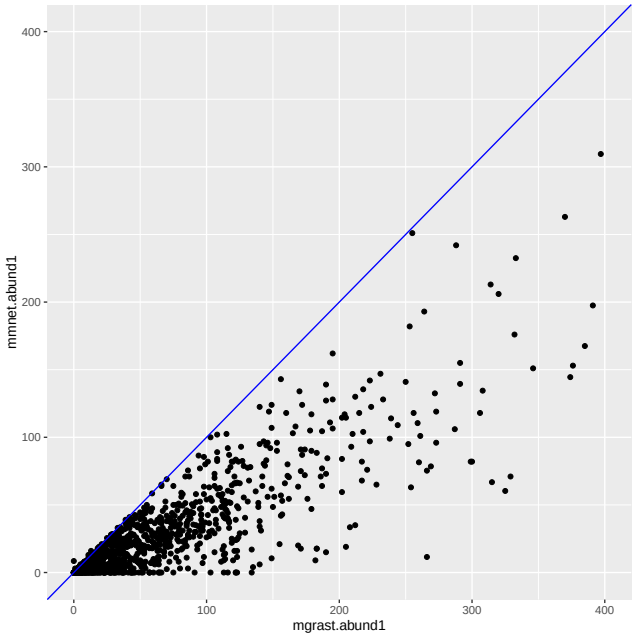

Figure 1: enzymatic genes abundance comparison between MG-RAST and mmnet package

among the top 100 matches; Sequences with multiple top KO-associated matches with the same e-value were annotated with the union set of KOs; To make a balance between identifying low-abundance genes and reducing the error-rate of identification a threshold of 2 reads to allow the inclusion of rare genes and all KO abundances below this threshold were set to zero.

For other tools, users can download the annotated data first. Taking IMG/M for example, metagenomic annotation data from IMG/M includes the abundance information. Thus, we can construct the metabolic network without abundance estimation directly. It can be accessed as following: 1) In package mmnet, function to construct metabolic network takes BIOM file as input. We create the BIOM file for network construction first:

```
## Load the IMG/M sample data
IMGFile <- system.file("extdata/IMGSample.tab", package = "mmnet")
IMGSample <- read.delim2(IMGFile, quote = "")
## Create BIOM file for network construction
abundance <- IMGSample$Gene.Count
abundance <- abundance/sum(abundance)
abundance <- as.data.frame(abundance)
KO <- IMGSample$KO.ID
KO <- as.data.frame(gsub("KO:", "", KO))
biom.data <- make_biom(abundance, observation_metadata = KO)
biom.data$type <- "enzymatic genes abundance"
```

2) Then we construct and analyze the SSN:

```
## Construct and analyze SSN
ssn <- constructSSN(biom.data)
topologicalAnalyzeNet(ssn)
```

Futhermore, users may have annotated the metagenomic reads locally, the annotated profile should be preprocessed in format as the MGRAST annotation profile that have 13 columns. Details:

1. Read id
2. Hit id, ref sequence id
3. percentage identity, e.g. 100.00
4. alignment length, e.g. 107
5. number of mismatches, e.g. 0
6. number of gap openings, e.g. 0
7. query start, e.g. 1
8. query .end, e.g. 107
9. hit start, e.g. 1262
10. hit .end, e.g. 1156
11. e-value, e.g. 1.7e-54
12. score in bits, e.g. 210.0
13. KEGG orthology

Then we can analyze the annotation profile properly as profile from MG-RAST. We also have added this description to the vignette of mmnet to help users analyze the annotation profiles from the other tools. And we still strive to improve our package with good support for other tools.

2.4 Building reference metabolic dataset

This package takes KEGG database to annotate enzymatic genes with metabolic reactions which is the basis representation for all proteins and functional RNAs corresponding to KEGG pathway nodes, BRITE hierarchy nodes, and KEGG module nodes to annotation the microbiome. The KEGG metabolic pathway is the best organized part of KEGG PATHWAY database, and also is a network of KO-KO relations (**?**). It is composed of KO, substrate and product of KO, and have been applied widely to create genome-scale metabolic networks of various microbial species (**?**).

This reference metabolic data was obtained with the KEGG free REST API, about 150 KEGG reference pathways. And KEGG API is provided for academic use by academic users belonging to academic institutions. This service should not be used for bulk data downloads. Thus, our small download with KEGG API is in agreement with the license.

An initial reference data named *RefDbcache.rda* was saved in "data" subdirectory of *mment* that contains KOs and annotated with a metabolic reaction. Moreover, an KO based reference metabolic network was also saved in *RefDbcache.rda*, where nodes represent enzymes (KOs), and the directed edge from enzyme A to enzyme B indicates that a product metabolite of a reaction catalyzed by enzyme A is a substrate metabolite of a reaction catalyzed by enzyme B. As KEGG database is constantly updated, reference data can be updated by function *updateMetabolicNetwork* manually and saved in the directory user specified.

Reference dataset can be loaded by function *loadMetabolicData*

```
loadMetabolicData()
summary(RefDbcache)

##           Length Class  Mode
## ko        3768   -none- character
## substrate 3768   -none- list
## product   3768   -none- list
## user         1   -none- character
## date         1   -none- character
## version     14   -none- list
## network      9   igraph list
```

2.5 Constructing State Specific Network

Different biological states (e.g. obese or lean) can be identified based on different enzymatic gene set and abundances. In network view, different biological states associates different metabolic sub-network with different abundances, named as State Specific Network (SSN). We provide function *constructSSN* to construct the reference network to obtain the state specific metabolic network for each microbiome. It could take the output of function *estimateAbundance* as input, give the SSN as output.

```
ssn <- constructSSN(mmnet.abund)
g <- ssn[[1]]
summary(g)

## IGRAPH DN-- 568 5229 -- SSN
## + attr: name (g/c), name (v/c), abundance (v/n)

abund <- get.vertex.attribute(g, "abundance", index = V(g))
summary(abund)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       2       8      25      39      49     310
```

2.6 Topological network analysis

To examine whether enzymes that are associated with a specific host state exhibit some topological features in the SSN, we provide function *topologicalAnalyzeNet* to compute and illustrate the correlations between the topological properties of enzymatic genes and their abundances. It links the difference abundance with the topological correlation (Figure 2). Common topological features are supported. See *'?topologicalAnalyzeNet'* for details.

```
topo.net <- topologicalAnalyzeNet(g)
## network with topological features as attributes
topo.net

## IGRAPH DN-- 568 5229 -- SSN
## + attr: name (g/c), name (v/c), abundance (v/n),
## | betweennessCentrality (v/x), degree (v/x),
## | clusteringCoefficient (v/x), pageRank (v/x)
```

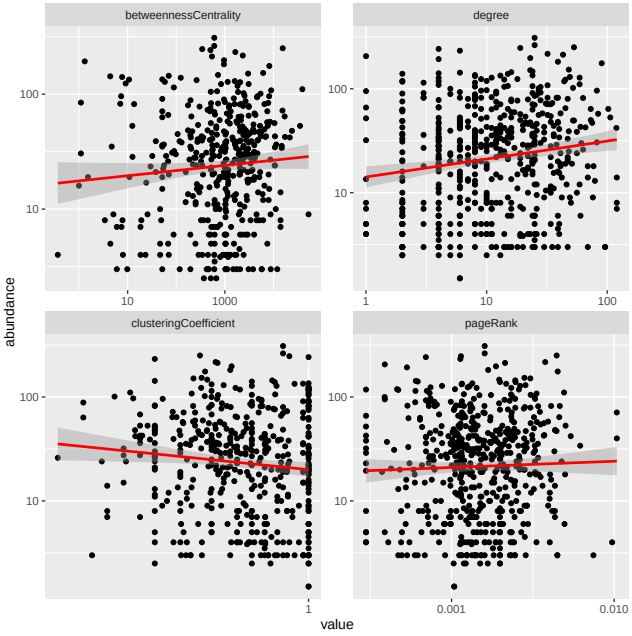

Figure 2: Topological metabolic network analysis, linking topological features and enzymatic gene abundances

```
## + edges (vertex names):
##  [1] K01623->K00128 K01623->K00134 K01623->K01803 K01623->K00850
##  [5] K01623->K01625 K01623->K01619 K01623->K00615 K01623->K00616
##  [9] K01623->K01629 K01623->K00895 K01623->K00057 K01623->K01734
## [13] K01623->K01632 K01623->K08681 K01623->K06215 K01623->K03517
## [17] K01623->K01624 K01623->K04041 K01623->K03856 K00128->K01623
## [21] K00128->K00382 K00128->K01895 K00128->K00174 K00128->K00658
## + ... omitted several edges
```

2.7 Differential network analysis

To compare the abundance of enzymatic genes across various samples, we take three strategies to identify the differential abundance (enrich or deplete), including (1) odds ratio, (2) difference rank and (3) Jensen-Shannon Divergence (JSD). The corresponding function *differentialAnalyzeNet* outputs the comparative network with nodes of differential abundance as result (Figure 3).

```
state <- c("obese", "lean")
differential.net <- differentialAnalyzeNet(ssn, sample.state = state,
    method = "OR", cutoff = c(0.5, 2))
summary(differential.net)

## IGRAPH DN-- 651 6512 -- refNet
## + attr: name (g/c), name (v/c), p.value (v/n), OR (v/n)
```

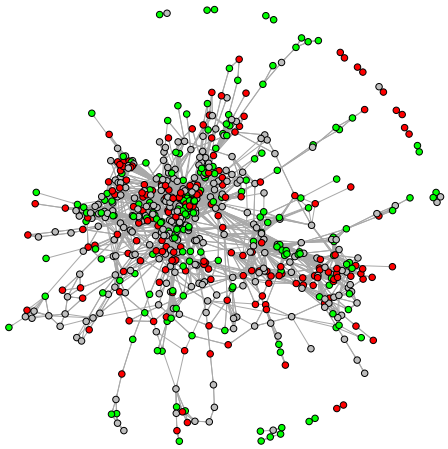

Figure 3: Differential metabolic network analysis, enzymatic genes that are associated with specific state appear as colored nodes (red=enriched, green=depleted)

2.8 Network Visualization

Both function *topologicalAnalyzeNet* and *differentialAnalyzeNet* utilize function *showMetagenomicNet* to plot the metabolic network. The showMetagenomicNet can also be used to personalized network display by specifying appropriate parameters (Figure 4).

```
## the reference network
## showMetagenomicNet(RefDbcache$network,mode='ref')
## the state specific metabolic network
showMetagenomicNet(g, mode = "ssn", vertex.label = NA,
    edge.width = 0.3, edge.arrow.size = 0.1, edge.arrow.width = 0.1,
    layout = layout.fruchterman.reingold)
```

3 Analysis in Cytoscape

Here is a simple example to show the reference metabolic network in Cytoscape with *RCytoscape* package (Figure 5). *Cytoscape2.8*, and *CytoscapeRPC*: a Cytoscape plugin is required besides of R package RCytoscape. See *RCytoscape* package for details on 'how to transfer the network and attributes from R to Cytoscape'.

Open Cytoscape, and then activate the CytoscapeRPC plugin in Cytoscape's Plugins menu. Click to start the XMLRPC server in the dialog, and Cytoscape will wait for commands from R. In the default setting, the communicate port is 9000 on localhost. You can choose a different port, just be sure to use that changed port number when you call the RCytoscape constructor.

Then, type the following commands in R:

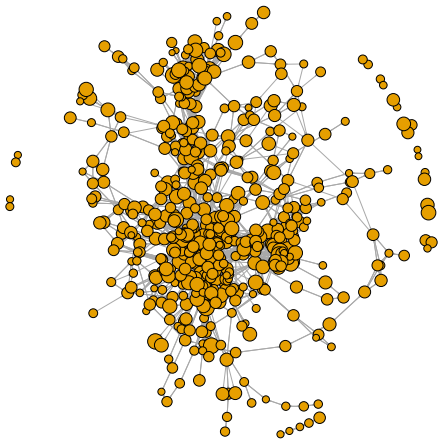

Figure 4: Visualization of State Specific Network using extitshowMetagenomicNet with node size proportional to log(abundance)

```
if (require(RCytoscape)) {
    refnet <- ssn[[1]]
    net <- igraph.to.graphNEL(refnet)
    ## initialize the edge attibute
    ## edge.attr=list.edge.attributes(refnet)
    ## edge.attr.class = sapply(edge.attr, class)
    ## edge.attr.class[edge.attr.class=='character']='char'
    ## init node attributes
    node.attr = list.vertex.attributes(refnet)
    if (length(node.attr)) {
        node.attr.class = sapply(node.attr, class)
        node.attr.class[node.attr.class == "character"] = "char"
        for (i in 1:length(node.attr)) net <- initNodeAttribute(net,
            attribute.name = node.attr[i], attribute.type = node.attr.class[i],
            default.value = "0")
    }
    ## our metagenomic network does not have edge
    ## attributes, set them all to 1
    net <- initEdgeAttribute(net, attribute.name = "weight",
        attribute.type = "numeric", default.value = "1")
    ## create a network window in Cytoscape
    cw <- new.CytoscapeWindow("net", graph = net, overwriteWindow = TRUE)
    ## transmits the CytoscapeWindowClass's graph data,
    ## from R to Cytoscape, nodes, edges, node and edge
    ## attributes
    displayGraph(cw)
```

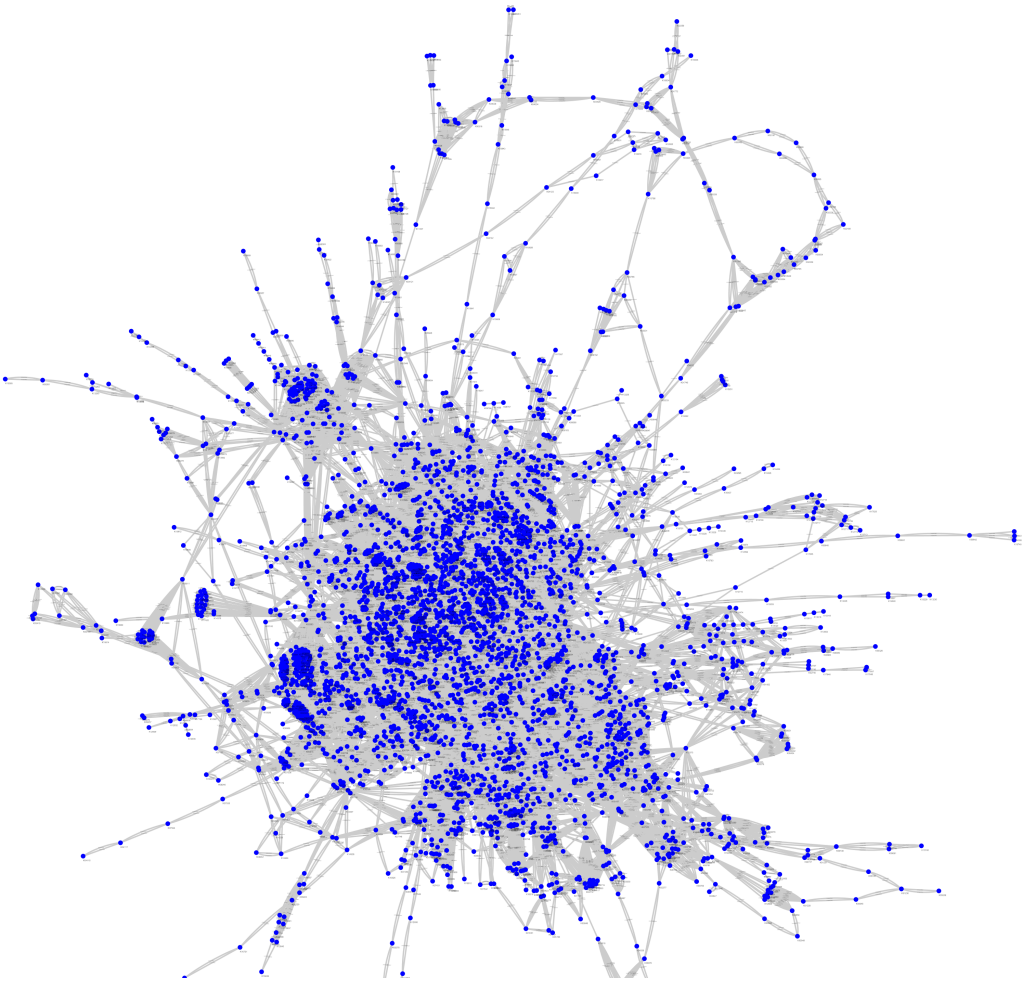

Figure 5: Export netowork to Cytoscape with *RCytoscape* package

```
}
```

Session Information

The version number of R and packages loaded for generating the vignette were:

```
## R version 3.2.3 (2015-12-10)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.9.5 (Mavericks)
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
```

```
## [1] stats     graphics  grDevices utils     datasets  methods
## [7] base
##
## other attached packages:
## [1] ggplot2_2.0.0  RCurl_1.95-4.7 bitops_1.0-6   mmnet_1.8.1
## [5] biom_0.3.12    igraph_1.0.1   knitr_1.12.3
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.3          formatR_1.2.1        plyr_1.8.3
##  [4] highr_0.5.1          XVector_0.10.0       tools_3.2.3
##  [7] zlibbioc_1.16.0      digest_0.6.9         evaluate_0.8
## [10] gtable_0.1.2         lattice_0.20-33      png_0.1-7
## [13] Matrix_1.2-3         parallel_3.2.3       httr_1.1.0
## [16] stringr_1.0.0        Biostrings_2.38.3    S4Vectors_0.8.11
## [19] IRanges_2.4.6        stats4_3.2.3         grid_3.2.3
## [22] nnet_7.3-11          Biobase_2.30.0       R6_2.1.2
## [25] flexmix_2.3-13       XML_3.98-1.3         RJSONIO_1.3-0
## [28] reshape2_1.4.1       magrittr_1.5         scales_0.3.0
## [31] codetools_0.2-14     modeltools_0.2-21    BiocGenerics_0.16.1
## [34] KEGGREST_1.10.1      colorspace_1.2-6     labeling_0.3
## [37] stringi_1.0-1        munsell_0.4.2
```

Cleanup

This is a cleanup step for the vignette on Windows; typically not needed for users.

```
allCon <- showConnections()
socketCon <- as.integer(rownames(allCon)[allCon[, "class"] ==
    "sockconn"])
sapply(socketCon, function(ii) close.connection(getConnection(ii)))

## list()
```