1 Introduction

Transcription factors (TFs), chromatin modifications and microRNAs (miRNAs) are important in regulating gene expression[1]. Chromatin immunoprecipitation (ChIP) followed by high-throughput sequencing (ChIP-seq)[2] or genome tiling array analysis (ChIP-chip)[3] are widely used technologies for identifying genome-wide binding sites of TFs (TFBDs)[4]. The role of the TF on genome-wide gene expression can be determined by expression microarray or RNA-seq experiments[5]. By combining both technologies, researchers have the potential to decipher the regulatory network of the TF. Genes bound by the TF and altered in expression are considered direct targets of the TF, and genes altered in expression but not bound by the TF are the indirect targets of the TF. The indirect targets can potentially form a complex network itself, especially when the TF is a master regulator who regulates other TFs. To facilitate identification of the complex regulatory network of TFs and how indirect targets are inter-connected, we have developed GeneNetworkBuilder (GNB). Each

---

*jianhong.ou@umassmed.edu
Julie.Zhu@umassmed.edu
generated network is consisted of a directed acyclic graph with each edge representing TF -> gene, TF -> miRNA, or miRNA -> gene where a -> b represents "a regulates b".

2 Examples of using GeneNetworkBuilder

To use GNB, users need to input a list of genes bound by a given TF and another list of genes/miRNAs with altered expression by knockdown/knockout of the same TF. The bound gene list could be obtained from ChIP-seq or ChIP-chip experiment. The gene list with altered expression are from RNA-seq or expression microarray experiment. ChIP experiments and expression experiments should preferably be performed in similar experimental condition such as tissue type, development stage etc. In addition, users need to select a TF regulatory network from GNB or upload customized TF regulatory network.

GNB provides two embedded regulatory networks. One is designed for Caenorhabditis elegans combining database EDGEdb[6] and microCosm Targets[7]. Database MicroCosm Targets contains computationally predicted targets for miRNAs across many species. EDGEdb contains experimentally determined interactions of ~934 worm TFs by high-throughput yeast on-hybrid (Y1H) assay. And the other embedded regulatory network is designed for Homo sapiens combining database FANTOM[8], miRGen[9] and microCosm Targets[7]. FANTOM stores physical interactions among the majority of human/mouse DNA-binding TFs. The miRGen is an integrated database of miRNA regulation by TFs and miRNA targets mainly for human. Figure 1 depicts the relationships of various databases and its role in GNB.

2.1 Quick start

Here is an example to use GNB to generate a simple regulatory network for C. elegans. There are three steps,

1. buildNetwork, build the network by GNB-embedded or user-defined regulatory network starting from the bound gene list.

2. filterNetwork, filter the network by differential expressed genes/miRNAs.

3. polishNetwork, generate the graphNEL object with display style.

```r
> ## load C. elegans miRNA ID lists
> data("ce.miRNA.map")
> ## load GNB-embedded regulatory network of C. elegans.
> data("ce.interactionmap")
> ## load data required
> data("example.data")
> ## build the network by binding list and interaction map
> sifNetwork<-buildNetwork(TFbindingTable=example.data$ce.bind,
+ interactionmap=ce.interactionmap, level=2)
> ## filter the network by expression data
> cifNetwork<-filterNetwork(rootgene="WBGene00000912", sifNetwork=sifNetwork,
+ exprsData=uniqueExprsData(example.data$ce.exprData),
+ level=2)
```
2.2 Example using gene expression profile

Here is an example to use GNB to generate a simple regulatory network for *C. elegans*. And also show some examples how to use the *graphNEL* object for further analysis.

> library(GeneNetworkBuilder)
> data("example.data")
> ##Initialize a binding matrix by TF and the related gene lists of TFEDs.
> ##For example, TF is *daf-16*, and the ChIP-chip result indicates that it can bind to
> ##upstream regions of gene "zip-2", "zip-4", "nhr-3" and "nhr-66".
> bind<-cbind(from="daf-16", to=c("zip-2", "zip-4", "nhr-3", "nhr-66"))
> ##For same gene, there are multiple gene aliases. In order to eliminate the possibility of
> ##missing any interactions, convert the gene symbols to unique gene ids is important.
data("ce.IDsMap")
bind<-convertID(toupper(bind), IDsMap=ce.IDsMap, ByName=c("from", "to"))
##build the network by binding list and interaction map
data("ce.interactionmap")
sifNetwork<-buildNetwork(TFbindingTable=example.data$ce.bind,
+ interactionmap=ce.interactionmap, level=2)
##filter the network by expression data
##For each gene id, it should have only single record for expression change.
unique.ce.microarrayData<-uniqueExprsData(example.data$ce.exprData,
+ method="Max", condenseName="logFC")
data("ce.miRNA.map")
cifNetwork<-filterNetwork(rootgene="WBGene00000912", sifNetwork=sifNetwork,
+ exprsData=unique.ce.microarrayData, mergeBy="symbols",
+ miRNAlist=as.character(ce.miRNA.map[,1]),
+ tolerance=1, cutoffFVal=0.01, cutoffLFO=1)
##convert the unique gene ids back to gene symbols
data("ce.mapIDs")
cifNetwork<-convertID(cifNetwork, ce.mapIDs, ByName=c("from", "to"))
##generate graphNEL object for the network
gR<-polishNetwork(cifNetwork, ce.mapIDs, ByName=c("from", "to"))
##plot the figure
library(Rgraphviz)
plotNetwork<-function(gR, layouttype="dot", ...){
  if(is(gR, "graphNEL")) stop("gR must be a graphNEL object")
  if(!GeneNetworkBuilder:::inList(layouttype, c("dot", "neato", "twopi", "circo", "fdp")))
    stop("layouttype must be dot, neato, twopi, circo or fdp")
  g1<-Rgraphviz::layoutGraph(gR, layoutType=layouttype, ...)
  nodeRenderInfo(g1)$col <- nodeRenderInfo(gR)$col
  nodeRenderInfo(g1)$fill <- nodeRenderInfo(gR)$fill
  renderGraph(g1)
}
plotNetwork(gR)
##output the GXL file
library("XML")
xml<-saveXML(toGXL(gR)$value())
z<-textConnection(xml)
cat(readLines(z, 8), sep="\n")

<?xml version="1.0"?>
<gxl:gxl xmlns:gxl="http://www.gupro.de/GXL/gxl-1.1.dtd"/>
<gxl:graph id="graphNEL" edgemode="directed">
  <gxl:node id="zip-4">
    <gxl:attr name="size">
      <gxl:float>72</gxl:float>
    </gxl:attr>
    <gxl:attr name="fill">
      <gxl:float>72</gxl:float>
    </gxl:attr>
  </gxl:node>
  <gxl:node id="daf-16">
    <gxl:attr name="size">
      <gxl:float>3</gxl:float>
    </gxl:attr>
  </gxl:node>
  <gxl:node id="lam-2">
    <gxl:attr name="size">
      <gxl:float>3</gxl:float>
    </gxl:attr>
  </gxl:node>
  <gxl:edge id="zip-4 -> daf-16" source="zip-4" target="daf-16">
    <gxl:attr name="length">
      <gxl:float>1</gxl:float>
    </gxl:attr>
  </gxl:edge>
  <gxl:edge id="daf-16 -> lam-2" source="daf-16" target="lam-2">
    <gxl:attr name="length">
      <gxl:float>1</gxl:float>
    </gxl:attr>
  </gxl:edge>
  <gxl:edge id="lam-2 -> zip-4" source="lam-2" target="zip-4">
    <gxl:attr name="length">
      <gxl:float>1</gxl:float>
    </gxl:attr>
  </gxl:edge>
</gxl:graph>

##calculate shortest path, ...
library(RBGL)
sp.between(gR,"daf-16","lam-2")

$'daf-16:lam-2'
$'daf-16:lam-2 $length
[1] 3

$'daf-16:lam-2' $path_detail
[1] "daf-16" "zip-4" "mir-46" "lam-2"

$'daf-16:lam-2 $length_detail
$'daf-16:lam-2 $length_detail[[1]]
daf-16->zip-4 zip-4->mir-46 mir-46->lam-2
1 1 1

4
2.3 Example using both gene and miRNA expression profile

Using several advanced genomic technologies including microarray profiling and miRNA sequencing, not only the gene expression profile, but also the miRNA expression profile can be obtained. A more robust network can be built if miRNA expression profiling is available. Here is an example to build SOX2 response network for *H. sapiens*. The data was downloaded from [BMC Genomics][10].

```r
library(GeneNetworkBuilder)
data("hs.interactionmap")
data("hs.miRNA.map")
data("hs.IDsMap")
data("hs.mapIDs")
data("example.data")
rootgene<-"6657"
sifNetwork<-buildNetwork(example.data$hs.bind, hs.interactionmap, level=5)
##example.data$ce$exprData is the combination of gene/miRNA expression profile
##note, here should set the miRNAtol to TRUE
```

Figure 2: Regulatory network of a *C. elegans* example
> cifNetwork<-filterNetwork(rootgene=rootgene, sifNetwork=sifNetwork,
+                     exprsData=example.data$hs.exprData, mergeBy="symbols",
+                     miRNAlist=as.character(hs.miRNA.map[,1]),
+                     tolerance=0, miRNAtol=TRUE)
> cifNetwork<-convertID(cifNetwork, hs.mapIDs, ByVal=c("from","to"))
> gR<-polishNetwork(cifNetwork)
#Plot the figure
> library(RCytoscape)
> gR <- initNodeAttribute(gR, "size", "numeric", 36)
> gR <- initNodeAttribute(gR, "fill", "char", "transparent")
> gR <- initEdgeAttribute(gR, "weight", "numeric", 1)
> cw <- new.CytoscapeWindow ("sox2", graph=gR)
> showGraphicsDetails (cw, TRUE)
> displayGraph(cw)
> layoutNetwork (cw, layout.name='force-directed')
> showGraphicsDetails(cw, T)
> rgb2hex <- function(x){
+     x<-col2rgb(x)
+     unlist(apply(x, 2, function(.ele)
+     +            rgb(.ele[1], .ele[2], .ele[3], maxColorValue=255))
+     +   )
+   }
> col <- rgb2hex(nodeRenderInfo(gR)$col)
> col <- col[!is.na(col)]
> fill <- rgb2hex(nodeRenderInfo(gR)$fill)
> fill <- fill[!is.na(fill)]
> for(i in 1:length(fill)){setNodeColorDirect(cw, names(fill)[i], fill[i])}
> for(i in 1:length(col)){setNodeBorderColorDirect(cw, names(col)[i], col[i])}
> data <- nodeData(gR)
> for(i in 1:length(data)){
+    setNodeSizeDirect(cw, names(data)[i], data[i]$size)
+  }
> redraw (cw)

3 References

References


[4] Chromatin immunoprecipitation (ChIP) of plant transcription factors followed by sequencing (ChIP-SEQ) or hybridization to whole genome arrays (ChIP-CHIP), Kerstin Kaufmann, Jose Muño, Magne østerås, Laurent Farinelli, Paweł Krajewski and Gerco C Angenent, Nature Protocols, 5: 457-472 (2010)

Figure 3: The SOX2 regulatory network.


[10] The SOX2 response program in glioblastoma multiforme: an integrated ChIP-seq, expression microarray, and microRNA analysis, Xuefeng Fang, Jae-Geun Yoon, Lisha li,
4 Session Info

> sessionInfo()

R version 3.3.1 (2016-06-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.1 LTS

locale:
[1] LC_CTYPE=en_US.UTF-8   LC_NUMERIC=C
[3] LC_TIME=en_US.UTF-8    LC_COLLATE=C
[5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8   LC_NAME=C
[9] LC_ADDRESS=C           LC_TELEPHONE=C

attached base packages:
[1] grid   parallel   stats   graphics  grDevices  utils   datasets
[8] methods base

other attached packages:
[1] RBGL_1.50.0            XML_3.98-1.4
[3] Rgraphviz_2.18.0       graph_1.52.0
[5] BiocGenerics_0.20.0    GeneNetworkBuilder_1.16.0
[7] Rcpp_0.12.7

loaded via a namespace (and not attached):
[1] plyr_1.8.4  tools_3.3.1  knitr_1.14  stats4_3.3.1