Example of NetworkHub: Ovarian cancer analysis with mTor pathway

Version: 1.1
Date: February 10, 2019
Developer: Hung-Ching Chang
Citation: Hung-Ching Chang, Chiao-Pei Chu, Shu-Ju Lin, Chuhsing Kate Hsiao (2019) NetworkHub: Prioritize hub gene node regulation with intra-network association

Outline

Package requirement

```
# (for pre-processing)
library("plyr")
library("XML")
# (for analysis)
library("igraph")
library("org.Hs.eg.db")
library("annotate")
```

Function requirement

(for pre-processing)
KGML_to_Matrix, tran2undirected, sub_expression_data, remove_missing_node
(for analysis)
NetworkHub, pathway_permute, rank_node

Global setting

Things need to be set up first!

- pathway name
- directory of KGML
  (The KGML file must be downloaded from the KEGG website into this folder!!!)
- directory of output files
- gene expression data: the data here are extracted from The Cancer Genome Atlas (TCGA) epithelial ovarian carcinoma. Only partial data are used in this document.
- column position of sample
- phenotype (ex: case & control)

```
# Pathway name
pathway_name <- "mTor"
# Directory of KGML
KGML_directory <- "D:/Ovarian cancer analysis (demo by mTor pathway)/hsa04150.xml"
# Directory for output files
save_directory <- "D:/Ovarian cancer analysis (demo by mTor pathway)/"
setwd(save_directory)
# Gene expression data
demo_data  <- read.csv("D:/Ovarian cancer analysis (demo by mTor pathway)/demo data (samp
le = 10).csv")
head(demo_data)

##   EntrezGeneID control.1 control.2 control.3 control.4 control.5   case.1
## 1           10  2.840332  2.961529  3.141170  3.000209  3.105509 2.744687
## 2          100  4.128707  4.427107  4.182850  5.392086  4.694693 5.004778
## 3         1000  6.845821  8.196124  5.568838  5.833900  5.153876 7.129954
## 4        10000  4.330266  4.943449  3.376370  4.273421  5.148195 4.669256
## 5    100008586  3.084185  3.160772  3.238897  2.992744  3.062233 2.941759
## 6        10001  5.041449  4.558348  4.350160  4.995861  4.345354 3.983882
##     case.2   case.3   case.4   case.5
## 1 3.105175 2.898894 2.844884 2.966170
## 2 5.209021 5.547467 4.885651 5.763047
## 3 6.705560 5.666397 5.402597 6.366199
## 4 4.178147 3.945790 3.497507 5.092796
## 5 2.994407 5.405756 3.028235 3.077371
## 6 4.473347 4.319935 4.849457 5.718568
```

Data include genetic information (EntrezGeneID) and gene expression.

```
# Starting & ending column (sample) of the gene expression data.
sample_start_pos <- 2
sample_end_pos <- 11
# Case & control phenotype
case_control <- rep(c(0,1),each = 5)
```

Pre-processing (Part 1)

Transfer KGML to undirected adjacency matrix

$(KGML \rightarrow directed\ adjacency\ matrix \rightarrow undirected\ adjacency\ matrix)$

```
# KGML -> directed adjacency matrix
KGML_to_Matrix(pathway_name = pathway_name, KGML_file_path = KGML_directory, save_path =
save_directory)

## [1] "Success"
```

If the imported KGML file is suitable for analysis, the function returns "Success".

```
# Directed adjacency matrix -> undirected adjacency matrix
load(paste0(pathway_name, "(directed).RData"))
adj_matrix <- tran2undirected(adj_matrix)
save(adj_matrix, file = paste0(pathway_name, "(undirected).RData"))
```

Pre-processing (part 2)

Identify the genes within the target pathway to retrieve the gene expression data from the input & create sub-dataset for following analysis

```
load(paste0(pathway_name, "(node_detail).RData"))
subdata <- sub_expression_data(gene_data = demo_data, node_detail = node_detail,
                               sample_start_pos = sample_start_pos, sample_end_pos = samp
le_end_pos, symbol_in_KGML = T)
pathway_node_expression <- t(subdata[[1]])
save(pathway_node_expression, file = paste0(pathway_name, "(node_expression).RData"))
missing_pos <- subdata[[2]]
```

The function **"sub_expression_data"** transforms the expression data from gene level to node level. The output of **"sub_expression_data"** is a list, called **"subdata"**. The first member of **"subdata"** is the node expression (matrix form) which should be transformed here because the following analyses require the matrix whose column represents gene and row represents sample. The second member of **"subdata"** is the position of nodes whose expression are missing. These missing nodes would be removed in part 3.

Pre-processing (part 3)

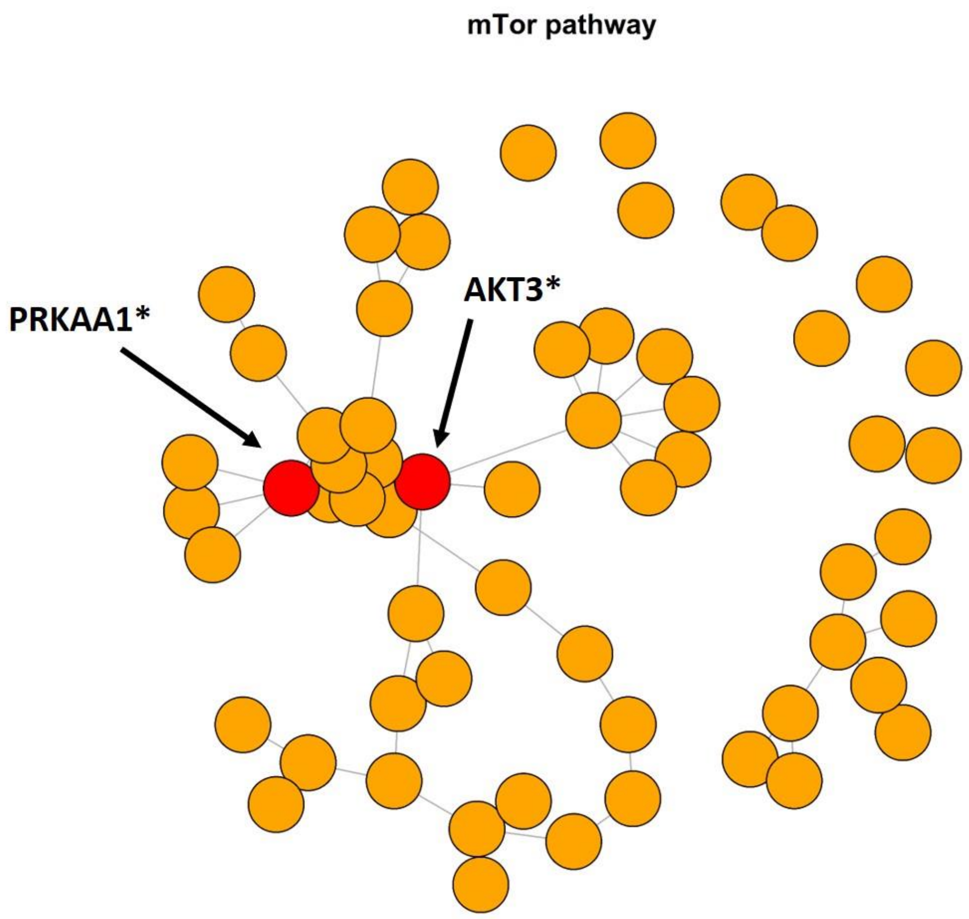
Remove the missing nodes in the "adjacency matrix" and "node_detail"

```
# load "undirected adjacency matrix" & "node_detail" files
load(paste0(pathway_name, "(undirected).RData"))
load(paste0(pathway_name, "(node_detail).RData"))
remaining_node <- remove_missing_node(missing_pos = missing_pos, adjacency_matrix = adj_m
atrix, node_detail_file = node_detail)
adj_matrix <- remaining_node[[1]]
save(adj_matrix, file = paste0(pathway_name, "(undirected).RData"))
node_detail <- remaining_node[[2]]
save(node_detail, file = paste0(pathway_name, "(node_detail).RData"))
```

The function **"remove_missing_node"** will remove the missing nodes in the adjacency matrix & node_detail (the indices of the nodes are saved in the variable **"missing_pos"**)
The **"remaining_node"** is also a list, just like **"subdata"**. The first member is the undirected adjacency matrix (no missing node) and the second member is node_detail (no missing node).

Pathway network

```
node_degree <-rowSums(adj_matrix)
two_hub_node <- which(rank(-node_degree) <= 2)
graph_network <- graph_from_adjacency_matrix(adj_matrix, mode = "undirected")
V(graph_network)$color <- "orange"
V(graph_network)$color[10] <- "red"
V(graph_network)$color[22] <- "blue"
V(graph_network)$label <- NA
E(graph_network)$edge.color <- "gray80"
plot(graph_network, vertex.label.cex = 1.5, main = paste0(pathway_name, " pathway"))
```



Analysis (Part 1)

Table of ranked nodes

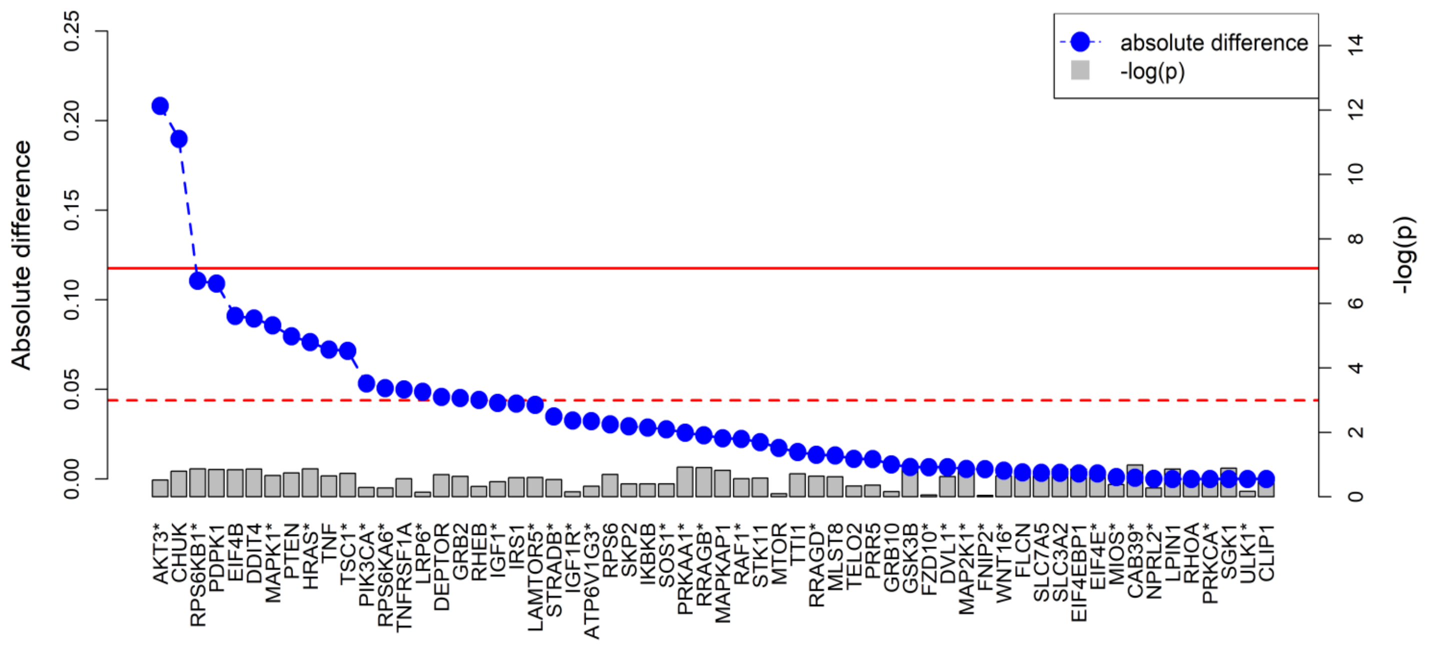
```
load(paste0(pathway_name, "(undirected).RData"))
load(paste0(pathway_name, "(node_expression).RData"))
node_degree <- rowSums(adj_matrix)
NetworkHub_stat <- rank_node(Node_expression = pathway_node_expression,
                             case_control = case_control,
                             adj_matrix = adj_matrix,
                             score_alpha = 1,
                             pvalue_alpha = 0.05)
pvalue <- NULL
for(ii in 1:dim(pathway_node_expression)[2]){
  pvalue <- c(pvalue, t.test(pathway_node_expression[,ii] ~ case_control)$p.value)
}
# Convert geneID into symbol
node_name <- names(NetworkHub_stat)
for(jj in 1:length(node_name)){
  temp_split <- strsplit(node_name[jj], "")[[1]]
  if(sum(temp_split == " ") > 0){
    start_pos <- which(temp_split == ":")[1] + 1
    end_pos <- which(temp_split == " ")[1] - 1
    ID <- paste(c(temp_split[start_pos:end_pos]), collapse = '')
    gene_symbol <- getSYMBOL(ID, data='org.Hs.eg')
    node_name[jj] <- paste(c(gene_symbol, "*"), collapse = '')
  }else{
    start_pos <- which(temp_split == ":")[1] + 1
    ID <- paste(c(temp_split[start_pos:length(temp_split)]), collapse = '')
    gene_symbol <- getSYMBOL(ID, data='org.Hs.eg')
    node_name[jj] <- paste(c(gene_symbol), collapse = '')
  }
}
rank_by_pvalue <- rank(pvalue)
result_table <- cbind(node_degree, NetworkHub_stat, pvalue, rank_by_pvalue)
rownames(result_table) <- node_name
colnames(result_table) <- c("node degree", "NetworkHub statistics", "t test", "rank by t
test")
node_rank_index <- order(-NetworkHub_stat)
result_table <- result_table[node_rank_index,]
head(result_table)

##          node degree NetworkHub statistics    t test rank by t test
## AKT3*             11            0.20820531 0.5251242             23
## CHUK               1            0.18976401 0.7969329             45
## RPS6KB1*           3            0.11065755 0.8645988             52
## PDPK1              3            0.10903068 0.8510450             48
## EIF4B              1            0.09101321 0.8404830             47
## DDIT4              8            0.08960545 0.8586135             49
```

Analysis (Part 2)

Plot of the ranked nodes

```r
par(mar=c(7, 5, 4, 5) + 0.1)
single_marker_pvalue <- result_table[,3]
names(single_marker_pvalue) <- rownames(result_table)
# First plot
df.bar <- barplot(single_marker_pvalue, col = "grey", ylim = c(0, 15), las = 2,
                  ylab = "", cex.lab = 1.25, axes = F)
abline(h = -log(0.05/dim(result_table)[1]), col = "red", lwd = 2)
abline(h = -log(0.05), col = "red", lwd = 2, lty = 2)
axis(side = 4)
mtext(side = 4, line = 3, "-log(p)", cex = 1.25)
# Second plot
xlim0 <- par()$usr[1:2]
par(new = T)
plot.new()
rank_stat <- result_table[,2]
plot.window(xlim = xlim0, ylim = c(0, max(rank_stat)*1.2), xaxs = "i")
lines(rank_stat ~ df.bar, lty = 2, lwd = 2, col = "blue")
points(rank_stat ~ df.bar, pch = 19, cex = 1.8, col = "blue", xaxt = "n")
axis(side = 2)
mtext(side = 2, line = 3, "Absolute difference", cex = 1.25)
legend("topright",
       legend=c("absolute difference", "-log(p)"),
       pch = c(19, 15), lty = c(2, NA), pt.cex = 2, ncol = 1, cex = 1.1,
       col=c("blue", "grey"))
```



Pathway test

This part is not important.

```
# Distance matrix (check symmetric)
diag(adj_matrix) <- 0
gg <- graph_from_adjacency_matrix(adj_matrix)
Distance_mat <- distances(gg)

pathway_stat <- NetworkHub(Node_expression = pathway_node_expression,
                                case_control = case_control,
                                Distance_mat = Distance_mat,
                                score_alpha = 1,
                                pvalue_alpha = 0.05)

pathway_test <- pathway_permute(replication = 1000,
                                Node_expression = pathway_node_expression,
                                case_control = case_control,
                                Distance_mat = Distance_mat,
                                score_alpha = 1,
                                pvalue_alpha = 0.05)
sum(pathway_test > pathway_stat)/1000   # p-value for pathway test

## [1] 0.854
```