richrr Updated files after adding puc compatibility to the gene microbe network cf07048 2 days ago

1 contributor

97 lines (76 sloc) 4.07 KB Raw Blame History

```
#library(TransNetDemo)
library(stringr)
library(ProNet)
library(igraph)


individualPvalueCutoff = 0.3
combinedPvalueCutoff = 0.05
combinedFDRCutoff = 0.2

# groups to compare
groupA = "HFHS"
groupB = "NCD"
sampleIdColName = "SampleID"
factorColName = "Factor"
geneSymbolColName ="IdSymbol"
# fold change column based on mean or median
foldchVar = "FoldChange_HFHS_NCD"   # or "FoldChange_Median_HFHS_NCD"

# map files
mapf1 = system.file("extdata", "mapping_file.rand.1.tsv", package = "TransNetDemo")
mapf2 = system.file("extdata", "mapping_file.rand.2.tsv", package = "TransNetDemo")

# gene files
genef1 = system.file("extdata", "gene_file_1.tsv", package = "TransNetDemo")
genef2 = system.file("extdata", "gene_file_2.tsv", package = "TransNetDemo")

# genes
Comp_genes = Compare_groups(mapf1, genef1, sampleIdColName, factorColName , groupA, groupB, geneSymbolColName)

# Select differentially abundant elements:
#########  (i) using a significance threshold #########
Sign_genes = Comp_genes[which(Comp_genes$FDR < 0.05),]
# this data has been saved as "Sign_genes_precomputed_1"
######### ######### ######### #########
# OR
######### (ii) in case you have multiple datasets, we recommend meta-analysis ###########
## we have already run these steps and stored the data under the variable "Sign_genes_metaanalysis_precomputed" ##
Comp_genes1 = Comp_genes
Comp_genes2 = Compare_groups(mapf2, genef2, sampleIdColName, factorColName , groupA, groupB, geneSymbolColName)
numbDatasets=2  # number of datasets

s_df = merge(Comp_genes1, Comp_genes2, by="row.names")
rownames(s_df) = rownames(Comp_genes1)
s_df = Check_consistency(s_df, foldchVar, 1, numbDatasets)
comb_in_df = Calc_combined(s_df)
Sign_genes = Apply_sign_cutoffs(comb_in_df, individualPvalueCutoff, combinedPvalueCutoff, combinedFDRCutoff )

# if this returns TRUE, you did everything correct!
identical(rownames(Sign_genes), rownames(Sign_genes_metaanalysis_precomputed))
#write.csv(Sign_genes,"Sign_genes_File.csv", quote=FALSE)
######### ######### ######### #########

```

```
# gene pairs
Corr_pairs = Correlation_in_group(mapf1, genef1, sampleIdColName, factorColName, groupA,  geneSymbolColName, rownames(Sign_genes))

# Select significant correlations:
#########  (i) using a significance threshold #########
Sign_pairs = Corr_pairs[which(Corr_pairs$pvalue < 0.05 & Corr_pairs$FDR < 0.1),]
######### ######### ######### #########
# OR
######### (ii) in case you have multiple datasets, we recommend meta-analysis ###########
## we have already run these steps and stored the data under the variable "Sign_genepairs_metaanalysis_precomputed" ##
Corr_pairs1 = Corr_pairs
Corr_pairs2 = Correlation_in_group(mapf2, genef2, sampleIdColName, factorColName, groupA,  geneSymbolColName, rownames(Sign_genes))

s_df = merge(Corr_pairs1, Corr_pairs2, by="row.names")
rownames(s_df) = rownames(Corr_pairs1)
s_df = Check_consistency(s_df, "Coefficient", 0, numbDatasets)
comb_in_df = Calc_combined(s_df)
Sign_pairs = Apply_sign_cutoffs(comb_in_df,individualPvalueCutoff, combinedPvalueCutoff, combinedFDRCutoff )

# if this returns TRUE, you did everything correct!
identical(rownames(Sign_pairs), rownames(Sign_genepairs_metaanalysis_precomputed))
#write.csv (Sign_pairs,"Sign_genes_pairs_File.csv", quote=FALSE)
######### ######### ######### #########


genes_df = Calc_median_val(Sign_genes, foldchVar)
identical(genes_df, Gene_df_precomputed)
pairs_df = Calc_median_val(Sign_pairs, "Coefficient")
outNetwork = Puc_compatible_network(pairs_df, genes_df)

# if this returns TRUE, you did everything correct!
identical(outNetwork[,c("partner1", "partner2")], Gene_network_precomputed[,c("partner1", "partner2")])
#write.csv (outNetwork,"gene-networkFile.csv", quote=FALSE)

cluster1 = Identify_subnetworks(outNetwork)
summary(cluster1)

# if this returns TRUE, you did everything correct!
identical(get.edgelist(cluster1), get.edgelist(genes_mcode_cluster1_precomputed)) # OR #sum(get.adjacency(cluster1) != get.adjacency(genes_
#write_graph(cluster1, "genes_mcode_cluster1_edges.txt", "ncol")

plot(cluster1, vertex.label=NA, layout= layout_in_circle, vertex.size=3)
```

richrr Updated files after adding puc compatibility to the gene microbe network cf07048 2 days ago

1 contributor

98 lines (76 sloc) 4.19 KB Raw Blame History

```
#library(TransNetDemo)
library(stringr)
library(ProNet)
library(igraph)


individualPvalueCutoff = 0.3
combinedPvalueCutoff = 0.05
combinedFDRCutoff = 0.2

# groups to compare
groupA = "HFHS"
groupB = "NCD"
sampleIdColName = "SampleID"
factorColName = "Factor"
microbeSymbolColName ="IdSymbol"
# fold change column based on mean or median
foldchVar = "FoldChange_HFHS_NCD"   # or "FoldChange_Median_HFHS_NCD"

# map files
mapf1 = system.file("extdata", "mapping_file.rand.1.tsv", package = "TransNetDemo")
mapf2 = system.file("extdata", "mapping_file.rand.2.tsv", package = "TransNetDemo")

# microbe files
microbef1 = system.file("extdata", "microbe_file_1.tsv", package = "TransNetDemo")
microbef2 = system.file("extdata", "microbe_file_2.tsv", package = "TransNetDemo")

# microbes
Comp_microbes = Compare_groups(mapf1, microbef1, sampleIdColName, factorColName , groupA, groupB, microbeSymbolColName)

# Select differentially abundant elements:
#########  (i) using a significance threshold #########
Sign_microbes = Comp_microbes[which(Comp_microbes$FDR < 0.05),]
# this data has been saved as "Sign_microbes_precomputed_1"
######### ######### ######### #########
# OR
######### (ii) in case you have multiple datasets, we recommend meta-analysis ###########
## we have already run these steps and stored the data under the variable "Sign_microbes_metaanalysis_precomputed" ##
Comp_microbes1 = Comp_microbes
Comp_microbes2 = Compare_groups(mapf2, microbef2, sampleIdColName, factorColName , groupA, groupB, microbeSymbolColName)
numbDatasets=2  # number of datasets

s_df = merge(Comp_microbes1, Comp_microbes2, by="row.names")
rownames(s_df) = rownames(Comp_microbes1)
s_df = Check_consistency(s_df, foldchVar, 1, numbDatasets)
comb_in_df = Calc_combined(s_df)
Sign_microbes = Apply_sign_cutoffs(comb_in_df, individualPvalueCutoff, combinedPvalueCutoff, combinedFDRCutoff )

# if this returns TRUE, you did everything correct!
identical(rownames(Sign_microbes), rownames(Sign_microbes_metaanalysis_precomputed))
#write.csv(Sign_microbes,"Sign_microbes_File.csv", quote=FALSE)
######### ######### ######### #########

```

```
# microbe pairs
Corr_pairs = Correlation_in_group(mapf1, microbef1, sampleIdColName, factorColName, groupA,  microbeSymbolColName, rownames(Sign_microbes))

# Select significant correlations:
#########  (i) using a significance threshold #########
Sign_pairs = Corr_pairs[which(Corr_pairs$pvalue < 0.05 & Corr_pairs$FDR < 0.1),]
######### ######### ######### #########
# OR
######### (ii) in case you have multiple datasets, we recommend meta-analysis ###########
## we have already run these steps and stored the data under the variable "Sign_microbepairs_metaanalysis_precomputed" ##
Corr_pairs1 = Corr_pairs
Corr_pairs2 = Correlation_in_group(mapf2, microbef2, sampleIdColName, factorColName, groupA,  microbeSymbolColName, rownames(Sign_microbes)

s_df = merge(Corr_pairs1, Corr_pairs2, by="row.names")
rownames(s_df) = rownames(Corr_pairs1)
s_df = Check_consistency(s_df, "Coefficient", 0, numbDatasets)
comb_in_df = Calc_combined(s_df)
Sign_pairs = Apply_sign_cutoffs(comb_in_df,individualPvalueCutoff, combinedPvalueCutoff, combinedFDRCutoff )

# if this returns TRUE, you did everything correct!
identical(rownames(Sign_pairs), rownames(Sign_microbepairs_metaanalysis_precomputed))
#write.csv (Sign_pairs,"Sign_microbes_pairs_File.csv", quote=FALSE)
######### ######### ######### #########


microbes_df = Calc_median_val(Sign_microbes, foldchVar)
identical(microbes_df, Microbe_df_precomputed)
pairs_df = Calc_median_val(Sign_pairs, "Coefficient")
outNetwork = Puc_compatible_network(pairs_df, microbes_df)

# if this returns TRUE, you did everything correct!
identical(outNetwork[,c("partner1", "partner2")], Microbe_network_precomputed[,c("partner1", "partner2")])
#write.csv (outNetwork,"microbe-networkFile.csv", quote=FALSE)

cluster1 = Identify_subnetworks(outNetwork)
summary(cluster1)

# if this returns TRUE, you did everything correct!
identical(get.edgelist(cluster1), get.edgelist(microbes_mcode_cluster1_precomputed)) # OR #sum(get.adjacency(cluster1) != get.adjacency(mic
#write_graph(cluster1, "microbes_mcode_cluster1_edges.txt", "ncol")

plot(cluster1, vertex.label=NA, vertex.size=3)

```

richrr added code to identify the top gene or microbe and label the importan… 97f87b8 2 days ago

1 contributor

156 lines (118 sloc) 5.74 KB Raw Blame History

```
#library(TransNetDemo)
library(stringr)
library(ProNet)
library(igraph)
library(ggplot2)


individualPvalueCutoff = 0.3
combinedPvalueCutoff = 0.05
combinedFDRCutoff = 0.2

# groups to compare
groupA = "HFHS"
groupB = "NCD"
sampleIdColName = "SampleID"
factorColName = "Factor"
gene_microbeSymbolColName ="IdSymbol"
# fold change column based on mean or median
foldchVar = "FoldChange_HFHS_NCD"   # or "FoldChange_Median_HFHS_NCD"

# map files
mapf1 = system.file("extdata", "mapping_file.rand.1.tsv", package = "TransNetDemo")
mapf2 = system.file("extdata", "mapping_file.rand.2.tsv", package = "TransNetDemo")

# gene_microbe files
gene_microbef1 = system.file("extdata", "gene_microbe_file_1.tsv", package = "TransNetDemo")
gene_microbef2 = system.file("extdata", "gene_microbe_file_2.tsv", package = "TransNetDemo")

pairs = expand.grid(V(genes_mcode_cluster1_precomputed)$name,V(microbes_mcode_cluster1_precomputed)$name)

# gene_microbe pairs
Corr_pairs = Correlation_in_group(mapf1, gene_microbef1, sampleIdColName, factorColName, groupA,  gene_microbeSymbolColName, NA, pairs)

# Select significant correlations:
#########  (i) using a significance threshold #########
Sign_pairs = Corr_pairs[which(Corr_pairs$pvalue < 0.05 & Corr_pairs$FDR < 0.1),]
######### ######### ######### #########
# OR
######### (ii) in case you have multiple datasets, we recommend meta-analysis ###########
## we have already run these steps and stored the data under the variable "Sign_gene_microbe_pairs_metaanalysis_precomputed" ##
Corr_pairs1 = Corr_pairs
Corr_pairs2 = Correlation_in_group(mapf2, gene_microbef2, sampleIdColName, factorColName, groupA,  gene_microbeSymbolColName, NA, pairs)
numbDatasets=2  # number of datasets

s_df = merge(Corr_pairs1, Corr_pairs2, by="row.names")
rownames(s_df) = rownames(Corr_pairs1)
s_df = Check_consistency(s_df, "Coefficient", 0, numbDatasets)
comb_in_df = Calc_combined(s_df)
Sign_pairs = Apply_sign_cutoffs(comb_in_df,individualPvalueCutoff, combinedPvalueCutoff, combinedFDRCutoff )

# if this returns TRUE, you did everything correct!
identical(Sign_pairs, Sign_gene_microbe_pairs_metaanalysis_precomputed)
#write.csv (Sign_pairs,"Sign_gene_microbes_pairs_File.csv", quote=FALSE)
```

```r
######### ######### ######### #########

genes_df = rbind(Gene_df_precomputed, Microbe_df_precomputed)
pairs_df = Calc_median_val(Sign_pairs, "Coefficient")
outNetwork = Puc_compatible_network(pairs_df, genes_df)
#write.csv (outNetwork,"gene_microbe-networkFile.csv", quote=FALSE)


g = graph_from_data_frame(outNetwork,directed = F, vertices = NULL)
# if this returns TRUE, you did everything correct!
identical(get.edgelist(g), get.edgelist(Gene_Microbe_network_precomputed))
#write_graph(g, "gene_microbe_edges.txt", "ncol")


# create TK (bipartite) network using:
# (i) genes_mcode_cluster1_precomputed
# (ii) microbes_mcode_cluster1_precomputed
# (iii) Gene_Microbe_network_precomputed


net1 = get.edgelist(genes_mcode_cluster1_precomputed)
head(net1)

net2 = get.edgelist(microbes_mcode_cluster1_precomputed)
head(net2)

net3 = get.edgelist(Gene_Microbe_network_precomputed)
head(net3)

TK_Network = graph_from_data_frame(rbind(net1, net2, net3), directed = F)
print(TK_Network, e=TRUE, v=TRUE)

# if this returns TRUE, you did everything correct!
identical(get.edgelist(TK_Network), get.edgelist(TK_Network_precomputed))
#write_graph(TK_Network, "Trans_Kingdom_NetworkFile.txt", "ncol")
#write_graph(TK_Network, "Trans_Kingdom_NetworkFile_indices.txt")


# write the mapping of nodes name to indices and group
nodes = data.frame()
for (vertex in V(TK_Network)) {
  Name = V(TK_Network)$name[vertex]
  Id = vertex
  Group = ""
  if(Name %in% as.vector(net1)){
    Group = "gene"
  } else {
    Group = "microbe"
  }
  nodes = rbind(nodes, cbind(Name, Id, Group))
}
colnames(nodes)
#write.table(nodes, "Trans_Kingdom_NetworkFile_nodes.txt", quote=F, row.names = F, sep=' ', col.names = T)

# calc bipartite betweenness centrality
# to find important genes
#FromNodes = as.numeric(nodes[nodes[,3]=="gene",2])
#ToNodes = as.numeric(nodes[nodes[,3]=="microbe",2])

# to find important microbes
FromNodes = as.numeric(nodes[nodes[,3]=="microbe",2])
ToNodes = as.numeric(nodes[nodes[,3]=="gene",2])

allPairs = expand.grid(FromNodes,ToNodes)
myNetwork = TK_Network
sumAllFractionsForAllNodes = Calc_bipartite_betweeness_centrality(allPairs, FromNodes, myNetwork)

head(sumAllFractionsForAllNodes)

# calculate node(s) with max. betweeness centrality
forPlot = colSums(sumAllFractionsForAllNodes)
topThree = sort(forPlot, decreasing = T)[1:3]
```

```
TopNode = as.integer(names(topThree[1]))

TopNodeName = nodes[TopNode, "Name"]

TK_Network = set_vertex_attr(TK_Network, "type", index = nodes$Id, as.factor(nodes$Group))
# no labels
#plot(TK_Network, vertex.label = NA, layout=layout_as_tree, vertex.color=c( "pink", "skyblue")[1+(V(TK_Network)$type==1)], vertex.size=4)

# label only the important node
plot(TK_Network, vertex.label = ifelse(V(TK_Network)$name==TopNodeName, V(TK_Network)$name, NA),
     layout=layout_as_tree, vertex.color=c( "pink", "skyblue")[1+(V(TK_Network)$type==1)], vertex.size=4, vertex.label.dist=0.15, vertex.la

# label only the microbe nodes
#plot(TK_Network, vertex.label = ifelse(nodes[which(nodes$Name==V(TK_Network)$name), "Group"]=='microbe', V(TK_Network)$name, NA),
#      layout=layout_as_tree, vertex.color=c( "pink", "skyblue")[1+(V(TK_Network)$type==1)], vertex.size=4, vertex.label.dist=0.15, vertex.

legend("topright",
       legend = unique(nodes$Group),
       col = c("skyblue" , "pink"),
       lty= 1,
       lwd = 5,
       cex=.7
)



# plot in two rows
#ltypes = c(TRUE, FALSE)[1+(V(TK_Network)$type==1)]
#plot(TK_Network, vertex.label = NA, vertex.size=3, layout=layout_as_bipartite(TK_Network,ltypes, hgap = 500), vertex.color=c( "pink", "sky
```

richrr finalized code with legends 643b5e8 19 days ago

1 contributor

66 lines (49 sloc) 1.92 KB Raw Blame History

```
#library(TransNetDemo)
library(gplots)


########################## extract inputs - study 1 ##############################

# groups to compare
groupA = "HFHS"
groupB = "NCD"
sampleIdColName = "SampleID"
factorColName = "Factor"
geneSymbolColName ="IdSymbol"

# map files
mapf1 = system.file("extdata", "mapping_file.rand.1.tsv", package = "TransNetDemo")

# read the mapping file
map1 = read.delim(mapf1, header=T)
rownames(map1) = map1[,sampleIdColName]

# select samples from groups
hfhs_samples1 = as.vector(map1[which(map1[,factorColName] == groupA),sampleIdColName])
ncd_samples1 = as.vector(map1[which(map1[,factorColName] == groupB),sampleIdColName])

# colors as per the group
samplecolors <- c(rep("magenta",25) , rep("blue",25))

# gene files
genef1 = system.file("extdata", "gene_file_1.tsv", package = "TransNetDemo")

# read the data file
genes1 = read.delim(genef1, header=T, check.names = F, row.names=1)

# make the heat map
dataset1=genes1[rownames(Sign_genes_precomputed_1), c(hfhs_samples1, ncd_samples1)]
dim(dataset1)
heatmap.2(as.matrix(dataset1), col=redgreen(75), ColSideColors=samplecolors, scale="row", key=TRUE, symkey=FALSE, density.info="none", trac

legend("topright",
       legend = unique(map1$Factor),
       col = c("magenta" , "blue"),
       lty= 1,
       lwd = 5,
       cex=.7
)

# microbe files
microbef1 = system.file("extdata", "microbe_file_1.tsv", package = "TransNetDemo")

# read the data file
microbes1 = read.delim(microbef1, header=T, check.names = F, row.names=1)

# make the heat map
```

```
dataset1=microbes1[rownames(Sign_microbes_precomputed_1), c(hfhs_samples1, ncd_samples1)]
dim(dataset1)
heatmap.2(as.matrix(dataset1), col=redgreen(75), ColSideColors=samplecolors, scale="row", key=TRUE, symkey=FALSE, density.info="none", trac

legend("topright",
       legend = unique(map1$Factor),
       col = c("magenta" , "blue"),
       lty= 1,
       lwd = 5,
       cex=.7
)

```

richrr Updated package f30fa7f 26 days ago

1 contributor

24 lines (22 sloc) 1.23 KB Raw Blame History

```r
#' Select significant elements
#'
#' This function applies the following significance cutoffs
#' individual p-value < 0.3; combined p-value < 0.05; fdr < 0.1
#'
#' @param df, individualPvalueCutoff, combinedPvalueCutoff, combinedFDRCutoff
#' @return A matrix with statistics for signifcant elements
#' @export
Apply_sign_cutoffs = function(df, individualPvalueCutoff = 0.3, combinedPvalueCutoff = 0.05, combinedFDRCutoff = 0.1){
  # find significant in individual pvalue: all of the pvalues for all of the datasets for each element must be smaller than threshold
  # find pvalue data
  # if it is a single dataset it becomes a vector so adding the drop=FALSE
  pvalueData = df[,grep("pvalue",colnames(df)), drop=FALSE]
  print(head(pvalueData))
  pvalueData = as.matrix(pvalueData)
  pvalueData = apply(pvalueData,2,function(x){as.numeric(as.vector(x))})

  # calculate the largest pvalue among all datasets for each gene, this largest pvalue must be smaller than threshold
  passIndevidualPvalue = apply(pvalueData,1,max)<individualPvalueCutoff
  Sign_elements = df[passIndevidualPvalue,]
  Sign_elements <- Sign_elements[Sign_elements$"combinedPvalue" < combinedPvalueCutoff  &  Sign_elements$"combinedFDR" < combinedFDRCutoff,
  return(Sign_elements)
}
```

richrr / TransNetDemo

Branch: master ▾ TransNetDemo / R / Calc_bipartite_betweeness_centrality.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

32 lines (26 sloc) 1020 Bytes Raw Blame History

```
#' Calc_bipartite_betweeness_centrality
#'
#' This function calculates bipartite_betweeness_centrality for each given element pair
#'
#' @param allPairs, FromNodes, myNetwork
#' @return A data frame with statistics for each element
#' @export
Calc_bipartite_betweeness_centrality = function(allPairs, FromNodes, myNetwork){

  sumAllFractionsForAllNodes = Get_template_matrix(FromNodes, allPairs)
  sumAllFractionsForAllNodes = as.data.frame(sumAllFractionsForAllNodes)
  #print(sumAllFractionsForAllNodes)
  counts = apply(as.matrix(allPairs),1,Get_shortest_paths,myNetwork)
  #print(counts)


   z = lapply(counts, function(x){
     if(nrow(x)>0){
       #print(x)
       columnsForUpdate = as.character(intersect(colnames(x),FromNodes))
       #print(columnsForUpdate)
       rowsForUpdate = rownames(x)
       #print(rowsForUpdate)
       #print(x[1, columnsForUpdate])
       sumAllFractionsForAllNodes[rowsForUpdate,columnsForUpdate] <<- x[1,columnsForUpdate]
     }
   })

  return(sumAllFractionsForAllNodes)
}

```

richrr / TransNetDemo

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Pulse Graphs

Branch: master ▾ TransNetDemo / R / Calc_combined.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

29 lines (26 sloc) 1.02 KB Raw Blame History

```
#' Calc combined p value across expts.
#'
#' This function code for calc combined p value across datasets.
#'
#' @param data frame
#' @return A matrix with combined pvalue and fdr
#' @export
Calc_combined = function(s_df){
  pvalueData = s_df[,grep("pvalue",colnames(s_df)), drop=FALSE]
  head(pvalueData)

  total_numb_input_files = ncol(pvalueData)
  interestedPvalueData = as.matrix(pvalueData)
  interestedPvalueData = apply(interestedPvalueData,2,function(x){as.numeric(as.vector(x))})
  combinedPvalue = apply(interestedPvalueData,1
                         ,function(pvalues){
                           pvalues = pvalues[!is.na(pvalues)]
                           statistics = -2*log(prod(pvalues))
                           degreeOfFreedom = 2*length(pvalues)
                           combined = 1-pchisq(statistics,degreeOfFreedom)
                         }
  )
  #calculate FDR for combined pvalue
  combinedFDR = p.adjust(combinedPvalue,method="fdr")
  comb_in_df = cbind(s_df, combinedPvalue, combinedFDR)

  return(comb_in_df)
}
```

richrr / TransNetDemo

Branch: master ▾ TransNetDemo / R / Calc_cor.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

19 lines (15 sloc) | 425 Bytes Raw Blame History

```
#' Calculate correlation between vectors
#'
#' This function calculate correlation between vectors
#'
#' @param pair, df, samples
#' @return A matrix with statistics for each element pair
#' @export
Calc_cor = function(pair, df, samples){
  idxs = as.vector(samples)

  c1 = as.numeric(df[pair[1], idxs])
  c2 = as.numeric(df[pair[2], idxs])

  p = cor.test(c1,c2)
  outLine = as.matrix(c(p$estimate,p$p.value))
  outLine
}

```

richrr / TransNetDemo

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Pulse Graphs

Branch: master ▾ TransNetDemo / R / Calc_median_val.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

20 lines (19 sloc) | 723 Bytes Raw Blame History

```r
#' Calculate median coefficient or fold change
#'
#' This function calculate median coefficient or fold change for each element
#'
#' @param df, pattern
#' @return A data frame with statistics for each element
#' @export
Calc_median_val = function(df, pattern){
  Colnames = colnames(df)[grep(pattern,colnames(df))]
  interestedData = df[,Colnames,drop=F]
  interestedData = as.matrix(interestedData)
  interestedData = apply(interestedData,2,function(x){as.numeric(as.vector(x))})
  combined = apply(interestedData,1, function(x){round(median(x, na.rm = TRUE), 3)})
  oldColnames = colnames(df)
  df = cbind(df,combined)
  colnames(df) = c(oldColnames, paste("combined", pattern, sep=''))
  print(head(df))
  return(df)
}
```

richrr / TransNetDemo

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Pulse Graphs

Branch: master TransNetDemo / R / Check_consistency.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

23 lines (19 sloc) 855 Bytes Raw Blame History

```r
#' Check consistent direction
#'
#' This function selects elements showing consistent direction across datasets
#'
#' @param df, patternToSearch, threshold
#' @return A matrix with elements showing consistent direction across datasets
#' @export
Check_consistency = function(s_df, pattern, Threshold, numbDatasets=2){
  # select the data of interest
  patternData = s_df[,grep(pattern,colnames(s_df)), drop=FALSE]
  head(patternData)

  rows_passing_consistency = c()
  res_pos = apply(patternData, 1, function(x) sum(x > Threshold))
  rows_passing_consistency = c(rows_passing_consistency, names(res_pos[res_pos==numbDatasets]))

  res_neg = apply(patternData, 1, function(x) sum(x < Threshold))
  rows_passing_consistency = c(rows_passing_consistency, names(res_neg[res_neg==numbDatasets]))

  s_df = s_df[rows_passing_consistency, -1]
  return(s_df)
}
```

Branch: master ▾ TransNetDemo / R / Compare_groups.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

45 lines (34 sloc) 1.69 KB Raw Blame History

```
#' Compare groups
#'
#' This function compares the values in two groups for each element
#'
#' @param mapFile, geneFile, sampleIdColName, factorColName, groupA, groupB, geneSymbolColName
#' @return A data frame with statistics for each element
#' @export
Compare_groups = function(mapFile, geneFile, sampleIdColName, factorColName, groupA, groupB, geneSymbolColName){

  ############################ extract inputs ###############################
  # read the mapping file
  map = read.delim(mapFile, header=T)
  rownames(map) = map[,sampleIdColName]
  head(map)

  # select samples from groups
  hfhs_samples = map[which(map[,factorColName] == groupA),]
  head(hfhs_samples)
  ncd_samples = map[which(map[,factorColName] == groupB),]
  head(ncd_samples)

  # read the data file
  genes = read.delim(geneFile, header=T, check.names = F)
  rownames(genes) = genes[,geneSymbolColName]
  head(genes)


  ############################ calc diff abundance ###############################
  out = sapply(rownames(genes), Diff_abundance, genes, rownames(hfhs_samples), rownames(ncd_samples))


  ############################ format output ###############################
  Comp_genes = t(out)
  colnames(Comp_genes) = c(paste("Mean", groupA, sep="_"), paste("Mean", groupB, sep="_"), paste("FoldChange", groupA, groupB, sep="_"),
                    paste("Median", groupA, sep="_"), paste("Median", groupB, sep="_"), paste("FoldChange","Median", groupA, groupB, sep="_


  ############################ calculate FDR  ############################
  FDR = p.adjust(Comp_genes[,"pvalue"],method="fdr")
  oldColnames = colnames(Comp_genes)
  Comp_genes = as.data.frame(cbind(Comp_genes, FDR))

  return(Comp_genes)
}
```

Branch: master ▾ TransNetDemo / R / Correlation_in_group.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

54 lines (42 sloc) 1.86 KB Raw Blame History

```
#' Correlation_in_group
#'
#' This function calculates correlation in a group for each element pair
#'
#' @param mapFile, geneFile, sampleIdColName, factorColName, groupA, geneSymbolColName, selected genes
#' @return A data frame with statistics for each element
#' @export
Correlation_in_group = function(mapFile, geneFile, sampleIdColName, factorColName, groupA,  geneSymbolColName, selected_genes, usepairs=NA){

  ########################### extract inputs ###############################
  # read the mapping file
  map = read.delim(mapFile, header=T)
  rownames(map) = map[,sampleIdColName]
  head(map)

  # select samples from group
  hfhs_samples = map[which(map[,factorColName] == groupA),]
  head(hfhs_samples)

  # read the data file
  genes = read.delim(geneFile, header=T, check.names = F)
  rownames(genes) = genes[,geneSymbolColName]
  head(genes)

  # create gene pairs if not already provided
  pairs = ''
  if(is.na(usepairs)) {
    pairs = t(combn(selected_genes,2))[,2:1]
  }else{
    pairs = usepairs
  }
  head(pairs)


  ############################## calculate correlation for each pair ###################################
  Corr_pairs = apply(pairs, 1, Calc_cor, genes, rownames(hfhs_samples))


  ############################## format output ###################################
  # add the pair as the column name
  colnames(Corr_pairs) = paste(as.vector(pairs[,1]),as.vector(pairs[,2]),sep="<==>")
  # the pairs become row names; and the corr coeff and pvalue are the column names
  Corr_pairs = t(Corr_pairs)
  # append method used to column labels
  colnames(Corr_pairs) = c(paste(groupA, "Coefficient",sep="_"), "pvalue")


  ############################## calculate FDR  ############################
  FDR = p.adjust(Corr_pairs[,colnames(Corr_pairs)[grep("pvalue",colnames(Corr_pairs))]],method="fdr")
  Corr_pairs = as.data.frame(cbind(Corr_pairs, FDR))

  return(Corr_pairs)
}
```

richrr / TransNetDemo

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Pulse Graphs

Branch: master ▾ TransNetDemo / R / Diff_abundance.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

29 lines (22 sloc) 790 Bytes Raw Blame History

```r
#' Compare abundance between groups for an element
#'
#' This function compares the values in two groups for an element
#'
#' @param element, df, samplesA, samplesB
#' @return A matrix with statistics for each element
#' @export
Diff_abundance = function(element, df, samplesA, samplesB){
  samplesA = as.vector(samplesA)
  samplesB = as.vector(samplesB)

  Vec1 = as.numeric(df[element, samplesA])
  Vec2 = as.numeric(df[element, samplesB])

  meanVec1 = mean(Vec1)
  meanVec2 = mean(Vec2)
  fold_change_mean = meanVec1/meanVec2

  medianVec1 = median(Vec1)
  medianVec2 = median(Vec2)
  fold_change_median = medianVec1/medianVec2

  p = t.test(Vec1, Vec2)

  Result = as.matrix(c(meanVec1, meanVec2, fold_change_mean, medianVec1 , medianVec2 , fold_change_median, p$p.value))

  Result
}
```

Branch: master TransNetDemo / R / Get_shortest_paths.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

34 lines (30 sloc) 1.58 KB Raw Blame History

```
#' Get counts for all shortest paths between given nodes in a bipartite networks
#'
#' This function calculates shortest paths between given nodes in a bipartite networks
#'
#' @param pair, myNetwork, sumAllFractionsForAllNodes
#' @return Updates the row in pair,sumAllFractionsForAllNodes
#' @export
Get_shortest_paths = function(pair,myNetwork){  # for each row in "sumAllFractionsForAllNodes", calculate the values for the columns and up
  #print(pair)
  allShortestPaths = get.all.shortest.paths(myNetwork, from= pair[1], to= pair[2], mode = "all", weights=NULL) # calculate the shortest pat
  #print(allShortestPaths)

  count = data.frame()
  if (length(allShortestPaths$res)==0){  # if there is no shortest path, do not update "sumAllFractionsForAllNodes"
    #fractions = cbind(FromNodes,rep(0,times=length(FromNodes)))
    #print("no paths")
  }else{
    allShortestPaths = do.call(rbind,allShortestPaths$res) # a matrix with each row containing a shortest path between the two nodes
    #print(allShortestPaths)

    nodesInPath = as.vector(allShortestPaths[,c(-1,-ncol(allShortestPaths))]) # get rid of the two nodes that are under study, the nodes in
    #print(nodesInPath)
    count = table(as.factor(nodesInPath))/nrow(allShortestPaths) # for each node in the shortest paths, calculate how many times(normalized
    #print(count)

    v = names(count)
    count = data.frame(rbind((count)))
    colnames(count) = v
    rownames(count) = paste(pair, collapse='_')
    #print(count)
  }
  count
}
```

richrr / TransNetDemo

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Pulse Graphs

Branch: master TransNetDemo / R / Get_template_matrix.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

22 lines (21 sloc) 1.05 KB Raw Blame History

```
#' Get_template_matrix
#'
#' This function creates a matrix , with each entry recording for each node(in a column),
#' if it appears(not 1 or 0, but a normalized number, since there can be more
#' than one shortest path between a pair of nodes) in the shortest path between a
#' pair of nodes(in a row), the rows contain all pairs of nodes between two groups.
#' Thus the sum of each column is the betweeness centrality for the node(in that column)
#'
#' @param FromNodes, allPairs
#' @return A matrix with rownames (source-target) and column names as nodes from source nodes set.
#' @export
Get_template_matrix = function(FromNodes, allPairs){
  sumAllFractionsForAllNodes = matrix(0,ncol=length(FromNodes)
                                      ,nrow=nrow(allPairs)
                                      ,dimnames= list(
                                        paste(as.vector(t(allPairs[,1])),as.vector(t(allPairs[,2])),sep="_")
                                        , as.character(FromNodes)
                                      )
  )
  return(sumAllFractionsForAllNodes)
}
```

Branch: master ▾ TransNetDemo / R / Identify_subnetworks.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

37 lines (30 sloc) | 1.16 KB Raw Blame History

```
#' Identify top subnetwork
#'
#' This function returns the induced subgraph of the nodes in the top cluster using mcode
#'
#'
#' @param network in which to identify subnetworks
#' @return A data frame (network) that is Puc compatiable
#' @export
Identify_subnetworks = function(outNetwork){

  # identify subnetworks
  dfNetwork = outNetwork[,c("partner1", "partner2", "combinedCoefficient")]
  print(head(dfNetwork))

  #?graph_from_data_frame
  # http://kateto.net/networks-r-igraph
  g = graph_from_data_frame(dfNetwork,directed = F, vertices = NULL)
  print(g, e=TRUE, v=TRUE)
  # see the edges and vertices
  #E(g) ; V(g) ; edge_attr(g) ; vertex_attr(g)

  # plots the networks
  #cluster(g,method="MCODE",layout="fruchterman.reingold")
  #cluster(g,method="FN",layout="fruchterman.reingold")

  #https://cran.r-project.org/web/packages/ProNet/vignettes/Tutorial.pdf
  # identify clusters
  result <- mcode(g,vwp=0.5,haircut=TRUE,fluff=FALSE,fdt=0.8,loops=FALSE)
  summary(result$COMPLEX)

  # plot the top cluster
  cluster1<-induced.subgraph(g,result$COMPLEX[[1]])
  #visualization(cluster1,node.size=4,node.label=V(cluster1)$name,node.label.color="blue")

  return(cluster1)
}
```

Branch: master ▾ TransNetDemo / R / Puc_compatiable_network.R Find file Copy path

richrr Updated package f30fa7f 26 days ago

1 contributor

83 lines (66 sloc) 3.69 KB Raw Blame History

```
#' Calculate network that satisfies the PUC criteria
#'
#' This function keeps edges that satisfy expected correlation and fold change relationships
#'
#'
#' @param pairs_df, genes_df
#' @return A data frame (network) that is Puc compatiable
#' @export
Puc_compatible_network = function(pairs_df, genes_df){
  #-------------------------------------------------------------------------------
  # calculate PUC
  #-------------------------------------------------------------------------------

  #change out format to partner1 partner2 for PUC
  row_names_pairs_df = rownames(pairs_df)
  head(row_names_pairs_df)
  pair = stringr::str_split( row_names_pairs_df ,"<==>")
  pairs = t(as.data.frame(pair))

  colnames(pairs) = c("partner1","partner2")
  pairs_df = apply(pairs_df, 2, function(x) as.numeric(as.character(x))) # convert the chars to numeric
  rownames(pairs) = row_names_pairs_df # remove this if you do not want row names
  outForPUC = cbind(pairs,pairs_df)
  head(outForPUC)


  # select pvalue columns
  grep_cols_c = grep("pvalue", colnames(outForPUC), value=TRUE, fixed=TRUE)
  # select "combinedPvalue", "combinedFDR", "combinedCoefficient"
  grep_cols_c = append(grep_cols_c, grep("combined" , colnames(outForPUC), value=TRUE, fixed=TRUE) )
  # select the partner columns
  g_grep_cols = c("partner1","partner2", grep_cols_c)

  outForPUC = outForPUC[,g_grep_cols]
  head(outForPUC)


  # attach the foldChange information for each partner
  FoldChangeCol = grep(paste("combined" , foldchVar, sep=''), colnames(genes_df), value=TRUE, fixed=TRUE)
  FoldMetab1_InPair = genes_df[as.vector(outForPUC[,"partner1"]), FoldChangeCol, drop=F]
  colnames(FoldMetab1_InPair) = c("partner1_FoldChange")
  FoldMetab2_InPair = genes_df[as.vector(outForPUC[,"partner2"]), FoldChangeCol, drop=F]
  colnames(FoldMetab2_InPair) = c("partner2_FoldChange")
  outForPUC = cbind(outForPUC,FoldMetab1_InPair,FoldMetab2_InPair)
  head(outForPUC)

  # calculate correlation Direction For combined correlation coefficient of interest
  # at this point we only have the consistent pairs left, so the value of combined corr coeff is ok to use
  interestedCoefficientColnames = grep("Coefficient",colnames(outForPUC), value=TRUE, fixed=TRUE)
  print(interestedCoefficientColnames)
  interestedCorrelationData = outForPUC[,interestedCoefficientColnames, drop=FALSE]
  interestedCorrelationData = apply(interestedCorrelationData,2,function(x){as.numeric(as.vector(x))})
  correlationDirection = interestedCorrelationData/abs(interestedCorrelationData)
```

```
54
55
56      # calculate fold change direction for each partner
57      FoldChangeColnames = colnames(outForPUC)[grep("FoldChange",colnames(outForPUC))] # since this is using the combined fold change calculate
58
59      FoldChangeData = outForPUC[,FoldChangeColnames]
60      FoldChangeDirection = (FoldChangeData-1)/abs(FoldChangeData-1)
61      names(FoldChangeDirection) = c()
62      colnames(FoldChangeDirection) = paste(colnames(FoldChangeData),"Direction",sep="_")
63
64      # calculate if fold change direction are the same for the two partners
65      IfFoldChangeDirectionMatch = apply(FoldChangeDirection,1,prod)
66      names(IfFoldChangeDirectionMatch) = c()
67      colnames(correlationDirection) = c()
68
69      # use "correlationDirection" and "IfFoldChangeDirectionMatch" to calc PUC,
70      # i.e. if these two are the same PUC=1 (good)
71      PUC = IfFoldChangeDirectionMatch * correlationDirection
72      outForPUC = cbind(outForPUC,correlationDirection,FoldChangeDirection,IfFoldChangeDirectionMatch,PUC)
73      head(outForPUC)
74
75      #write.csv(outForPUC, "PUC-output.csv" ,row.names=FALSE)
76
77      # find PUC expected
78      out = outForPUC[outForPUC[,"PUC"]==1 ,]
79      outNetwork = out[!is.na(out$"PUC"),] # remove the rows with 'NA' in PUC columns
80      return(outNetwork)
81
82  }
```