HTSanalyzeR2: An R package for functional annotation, network analysis and time series analysis of high-throughput data

***Lina Zhu*[1], *Feng Gao*[1], *Xiupei Mei*[2], *and Xin Wang*[1]**

[1]Department of Biomedical Sciences, City University of Hong Kong, Hong Kong
[2]Department of Computer Science, City University of Hong Kong, Hong Kong

2019-02-16

Abstract

This package provides gene set analysis, enriched subnetwork analyses and 'Time-series' functional analysis for various preprocessed high-throughput data generated either by CRISPR screening, RNA-seq, micro-array or RNAi in a unified workflow. More importantly, it could generate an interactive shiny report encompassing all the results and visualizations, facilitating the users maximally for downloading, modifying the visualization parts with personal preference and sharing with others by publishing the report to Shinyapps.io.

Package

HTSanalyzeR2 0.99.15

Contents

1 An overview of HTSanalyzeR2

Diverse high-throughput technologies such as microarray, RNA-seq, RNAi and CRISPR bring a huge potential to investigate the underlying biological mechanism under a specific condition, yet also cause great inconvenience for researchers to efficiently analyze such diverse data in a unified workflow. There is also no software so far claimed to be able to perform functional annotation for time series data with interactive visualization. Here, we have implemented a versatile R package, **HTSanalyzeR2**, which has several advantages as below:

- **HTSanalyzeR2** can perform gene set analysis and network analyses for pre-processed data generated by various popular high-throughput experiments including RNA-seq, CRISPR, micro-array and RNAi in a unified workflow.
- For time series data or the same experiment coming from different research groups, **HTSanalyzeR2** can perform time series analysis and comparative analysis for better mutual comparison.
- **HTSanalyzeR2** could generate an interactive report for users downloading, modifying the visualizations as well as sharing with others.

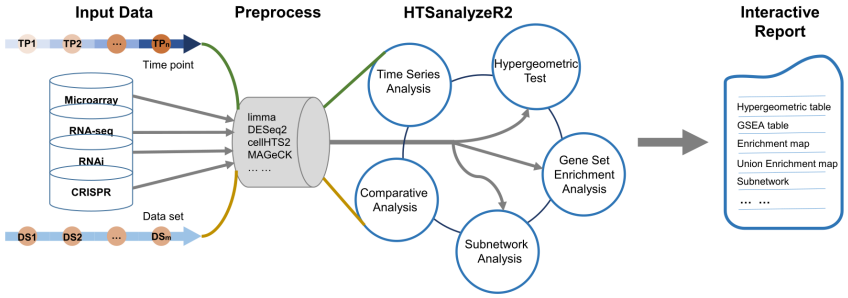

Figure 1: A schematic workflow of HTSanalyzeR2

1.1 Supported analysis

- Gene set over-representation analysis (GSOA): Measure the significance of overlap between user's interested genes (hits) and gene sets by hypergeometric test.
- Gene set enrichment analysis (GSEA): Measure the concordant trend of a gene set in one phenotype.
- Enriched subnetwork analysis: Identify subnetworks enriched for genes highly associated with the studied phenotype given a known network.
- Time series analysis/Comparative analysis: All aboved analysis on time-course data sets with several time points or multiple data sets with the same phenotype from different groups, by which to better compare the gene functional annotation results among different time points or data sets from different groups.

1.2 Supported data types

- Interested gene list
- Named phenotypes preprocessed from either **RNA-seq**, **micro-array**, **RNAi** or **CRISPR**
- **'Time Series' data**

1.3 Supported ontologies/pathways

- Gene Ontology: GO
  - Molecular function (MF)
  - Biological process (BP)
  - Cellular component (CC)
- Kyoto Encyclopedia of Genes and Genomes pathways: KEGG
- Molecular Signatures Database: MSigDB v6.1
  - h: hallmark gene sets
  - c1: positional gene sets
  - c2: curated gene sets
  - c3: motif gene sets
  - c4: computational gene sets
  - c5: GO gene sets
  - c6: oncogenic signatures
  - c7: immunologic signatures
- Customized gene sets

1.4 Supported species

- Gene Ontology and KEGG gene sets support any species that have an **OrgDb** object in Bioconductor.
- MSigDB gene sets support all 8 gene set collections for **Homo Sapiens** and three gene set collections: 'c2', 'c6' and 'c7' for **Mus musculus**.

1.5 Visualization

- GSEA plot
- Enrichment map
- Enriched subnetwork
- Interactive report

Before starting the demonstration, you need to install and load the following packages:

```
library(HTSanalyzeR2)
library(org.Hs.eg.db)
library(KEGGREST)
library(GO.db)
library(igraph)
library(limma)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
```

2 Case study1: Single dataset analysis with gene expression data

This case study uses **HTSanalyzeR2** to perform gene set over-representation analysis (GSOA), gene set enrichment analysis (GSEA) and enriched subnetwork analyses on a common gene expression profile. Basically, this dataset is from a micro-array experiment on 90 colon cancer patients with GEO number named GSE33113. Using the Colon Cancer Consensus Molecular Subtyping classifier generated by Guinney J et al. in 2015 (Guinney J (2015)), we can easily get the subtype label of each patient. Motivated by the poorest prognosis of CMS4 patients, we want to detect the enriched pathways of CMS4 patients compared to non-CMS4 patients. To this end, first we need to do the differential expression analysis using the most popular R packge 'limma' tailored for micro-array data.

2.1 microarray data preprocessing by 'limma'

```
data(GSE33113_exp)
data(GSE33113_label)

## delete samples with no CMS label
GSE33113_label <- GSE33113_label[which(!is.na(GSE33113_label))]
GSE33113_exp <- GSE33113_exp[, match(names(GSE33113_label),
                                     colnames(GSE33113_exp))]

## differential expression analysis using "limma" package between
## CMS4 samples and other samples
group <- rep(0, ncol(GSE33113_exp))
group[which(GSE33113_label == "CMS4")] <- 1

fit <- lmFit(GSE33113_exp, model.matrix(~ group))
fit <- eBayes(fit)
GSE33113_limma <- topTable(fit, coef=2, number=Inf, adjust.method="BH")
```

2.2 Gene set over-representation analysis (GSOA) and gene set enrichment analysis (GSEA)

2.2.1 Prepare the input data

To perform GSEA for single dataset, you must preprare the following inputs:

1. a named numeric vector of phenotypes (usually this would be a vector of genes with log2 fold change).
2. a list of gene set collections (could be generated by **HTSanalyzeR2** or use customized gene sets).

First you need to prepare a named phenotype.

```
phenotype <- as.vector(GSE33113_limma$logFC)
names(phenotype) <- rownames(GSE33113_limma)
```

Then, if you also want to do GSOA on a list of interested genes by hypergeometric test, you need to define the 'hits' as your interested genes. For example, here we define the hits as genes with absolute log2 fold change greater than 1 and adjust *p* value less than 0.05. **In this case, the names of phenotype, namely all the input genes, would be taken as the background gene list to perform hypergeometric test.**

Note:In cases if you want to do GSOA with only a list of hits and no background, **HTSanalyzeR2** can also realize it. For details please go to Part5: Special usage of HTSanalyzeR2.

```
## define hits if you want to do GSOA
hits <- rownames(GSE33113_limma[abs(GSE33113_limma$logFC) > 1 &
                                GSE33113_limma$adj.P.Val < 0.05, ])
```

Then we must define the gene set collections. A gene set collection is a list of gene sets, each of which consists of a group of genes with the same known function. **HTSanalyzeR2** provides facilities which greatly simplify the creation of up-to-date gene set collections including three Gene Ontology terms: Molecular Function (MF), Biological Process (BP), Cellular Component (CC) and KEGG pathways. Gene sets in a comprehensive molecular signatures database, MSigDB(Arthur Liberzon (2011)), for Homo Sapiens and Mus musculus are also provided. Here, to simplify the demonstration, we will only use one GO, KEGG and one MSigDB gene set collection. To work properly, you need to choose the right species for your input genes. Besides, these gene set collections must be provided as a named list as below:

```
## generate gene set collection
GO_MF <- GOGeneSets(species="Hs", ontologies=c("MF"))
PW_KEGG <- KeggGeneSets(species="Hs")
MSig_C2 <- MSigDBGeneSets(collection = "c2", species = "Hs")

## combine all needed gene set collections into a named list for further analysis
ListGSC <- list(GO_MF=GO_MF, PW_KEGG=PW_KEGG, MSig_C2=MSig_C2)
```

2.2.2 Initialize and preprocess

An S4 class named 'GSCA' is developed to perform GSOA in order to find the gene sets sharing significant overlapping with hits. Gene set enrichment analysis (GSEA), as described by Subramanian et al. (Subramanian A (2005)), can also be conducted.

To initialize a new 'GSCA' object, the previous prepared phenotype and a named list of gene sets collections are needed. In addition, as said before, if you also want to do GSOA, 'hits' is needed.

```
gsca <- GSCA(listOfGeneSetCollections=ListGSC,
             geneList=phenotype, hits=hits)
```

Then a preprocess step including invalid input data removing, duplication removing by different methods, initial gene identifiers converting to Entrez ID and phenotype ordering needs to be performed to fit for the next analysis. See the help documentation of funciton *preprocess* for more details.

```
gsca1 <- preprocess(gsca, species="Hs", initialIDs="SYMBOL",
                    keepMultipleMappings=TRUE, duplicateRemoverMethod="max",
                    orderAbsValue=FALSE)
```

2.2.3 Perform analysis

After getting a preprocessed 'GSCA' object, you can perform gene set over-representation analysis (GSOA) and gene set enrichment analysis (GSEA) using the function named *analyze*. This function needs an argument called *para*, which is a list of parameters including:

- *pValueCutoff*: a single numeric value specifying the cutoff for adjusted pvalues considered significant.
- *pAdjustMethod*: a single character value specifying the pvalue adjustment method.
- *nPermutations*: a single numeric value specifying the number of permutations for deriving p-values of GSEA.
- *minGeneSetSize*: a single numeric value specifying the minimum number of genes shared by a gene set and the background genes, namely the phenotype. Gene sets with fewer than this number are removed from both GSOA and GSEA.
- *exponent*: a single integer or numeric value used in weighting phenotypes in GSEA, as described by Subramanian et al. (Subramanian A (2005)).

```
gsca2 <- analyze(gsca1,
                 para=list(pValueCutoff=0.05, pAdjustMethod="BH",
                           nPermutations=100, minGeneSetSize=150,
                           exponent=1),
                 doGSOA = TRUE, doGSEA = TRUE)
```

In this case study, we only use 100 permutations and set a relative large *minGeneSetSize* just for a fast compilation of this vignette. In real applications, you may want a much smaller threshold (e.g. 10) and more permutation times (e.g. 10000) to get a more meaningful GSEA result.

During the enrichment analysis of gene sets, the function evaluates the statistical significance of the gene set scores by performing a large number of permutations. To analyze it more efficiently, our package allows parallel calculation based on the *doParallel* package. To do this, the user simply needs to register and claim to use multiple cores **before** running *analyze*.

```
## analyze using 4 cores
if (requireNamespace("doParallel", quietly=TRUE)) {
    doParallel::registerDoParallel(cores=4)
} else {
}

gsca2 <- analyze(gsca1,
                 para=list(pValueCutoff=0.05, pAdjustMethod="BH",
                           nPermutations=100, minGeneSetSize=150,
                           exponent=1),
                 doGSOA = TRUE, doGSEA = TRUE)
```

After analyzing, all the results are stored in slot *result* and can be easily accessed using a function named *getResult*. If gene GSOA and GSEA are both performed, gene sets which are both significant in this two analysis based on either pvalue or adjusted pvalue can be accessed.

```
## 1. GSOA result using MF gene sets from GO
head(getResult(gsca2)$HyperGeo.results$GO_MF, 3)
##            Universe Size Gene Set Size Total Hits Expected Hits
## GO:0008201         18757           151        450      3.622648
## GO:0005509         18757           622        450     14.922429
## GO:0008083         18757           152        450      3.646639
##            Observed Hits       Pvalue Adjusted.Pvalue
## GO:0008201            32 2.085573e-21    4.204784e-20
## GO:0005509            40 1.620863e-08    1.220352e-07
## GO:0008083            14 1.826308e-05    9.591952e-05

## 2. GSEA result using KEGG gene sets
head(getResult(gsca2)$GSEA.results$PW_KEGG, 3)
##          Observed.score Pvalue Adjusted.Pvalue
## hsa04310      0.5214548      0               0
## hsa04022      0.4889007      0               0
## hsa05152      0.5273596      0               0

## 3. result both significant regarding to pvalues in GSOA
##    and GSEA using 'c2' gene sets from MSigDB
head(getResult(gsca2)$Sig.pvals.in.both$MSig_C2, 3)
##                                                    HyperGeo.Pvalue
## TONKS_TARGETS_OF_RUNX1_RUNX1T1_FUSION_ERYTHROCYTE_UP    1.571212e-05
## TAKEDA_TARGETS_OF_NUP98_HOXA9_FUSION_8D_UP              1.694532e-05
## BROWNE_HCMV_INFECTION_6HR_DN                            3.281123e-04
##                                                    GSEA.Pvalue
## TONKS_TARGETS_OF_RUNX1_RUNX1T1_FUSION_ERYTHROCYTE_UP           0
## TAKEDA_TARGETS_OF_NUP98_HOXA9_FUSION_8D_UP                     0
## BROWNE_HCMV_INFECTION_6HR_DN                                   0

## 4. result both significant regarding to adjust pvalues in GSOA
##    and GSEA using 'c2' gene sets from MSigDB
head(getResult(gsca2)$Sig.adj.pvals.in.both$MSig_C2, 3)
##                                                    HyperGeo.Adj.Pvalue
## TONKS_TARGETS_OF_RUNX1_RUNX1T1_FUSION_ERYTHROCYTE_UP        8.465580e-05
## TAKEDA_TARGETS_OF_NUP98_HOXA9_FUSION_8D_UP                  9.051988e-05
## BROWNE_HCMV_INFECTION_6HR_DN                                1.362619e-03
##                                                    GSEA.Adj.Pvalue
## TONKS_TARGETS_OF_RUNX1_RUNX1T1_FUSION_ERYTHROCYTE_UP               0
## TAKEDA_TARGETS_OF_NUP98_HOXA9_FUSION_8D_UP                         0
## BROWNE_HCMV_INFECTION_6HR_DN                                       0
```

In addition, to make the results more understandable, users are highly recommended to annotate the gene sets ID to names by function *appendGSTerms*. As a result, an additional column named 'Gene.Set.Term' would appear.

```
gsca3 <- appendGSTerms(gsca2, goGSCs=c("GO_MF"),
                       keggGSCs=c("PW_KEGG"),
                       msigdbGSCs = c("MSig_C2"))

head(getResult(gsca3)$GSEA.results$PW_KEGG, 3)
```

```
##                       Gene.Set.Term Observed.score Pvalue Adjusted.Pvalue
## hsa04310      Wnt signaling pathway      0.5214548      0               0
## hsa04022 cGMP-PKG signaling pathway      0.4889007      0               0
## hsa05152               Tuberculosis      0.5273596      0               0
```

2.2.4 Summarize results

A *summarize* method could be performed to get a general summary for an analyzed 'GSCA' object including the gene set collections, genelist, hits, parameters for analysis and the summary of result.

```
summarize(gsca3)
##
## -No of genes in Gene set collections:
##         input above min size
## GO_MF    4182             53
## PW_KEGG   330             40
## MSig_C2  3762            532
##
##
## -No of genes in Gene List:
##           input valid duplicate removed converted to entrez
## Gene List 21656 21655             21655               18757
##
##
## -No of hits:
##      input preprocessed
## Hits   469          450
##
##
## -Parameters for analysis:
##               minGeneSetSize pValueCutoff pAdjustMethod
## HyperGeo Test 150            0.05         BH
##
##      minGeneSetSize pValueCutoff pAdjustMethod nPermutations exponent
## GSEA 150            0.05         BH            100           1
##
##
## -Significant gene sets (adjusted p-value< 0.05 ):
##          GO_MF PW_KEGG MSig_C2
## HyperGeo     8       8     223
## GSEA        19      22     398
## Both         7       8     204
```

2.2.5 Plot gene sets

To better view the GSEA result for a single gene set, you can use *viewGSEA* to plot the positions of the genes of the gene set in the ranked phenotypes and the location of the enrichment score. To this end, you must first get the gene set ID by *getTopGeneSets*, which

can return all or the top significant gene sets from GSEA results. Basically, the user needs to specify the type of results – "HyperGeo.results" or "GSEA.results", the name(s) of the gene set collection(s) as well as the type of selection– all (by parameter 'allSig') or top (by parameter 'ntop') significant gene sets.

```
topGS <- getTopGeneSets(gsca3, resultName="GSEA.results",
                        gscs=c("GO_MF", "PW_KEGG"), allSig=TRUE)
topGS
## $GO_MF
##   GO:0008201   GO:0008083   GO:0016887   GO:0005125   GO:0004252
## "GO:0008201" "GO:0008083" "GO:0016887" "GO:0005125" "GO:0004252"
##   GO:0004888   GO:0038023   GO:0000287   GO:0005096   GO:0003779
## "GO:0004888" "GO:0038023" "GO:0000287" "GO:0005096" "GO:0003779"
##   GO:0004930   GO:0005102   GO:0005509   GO:0003723   GO:0005524
## "GO:0004930" "GO:0005102" "GO:0005509" "GO:0003723" "GO:0005524"
##   GO:0000981   GO:0005515   GO:0003682   GO:0042803
## "GO:0000981" "GO:0005515" "GO:0003682" "GO:0042803"
##
## $PW_KEGG
##   hsa04310   hsa04022   hsa05152   hsa05016   hsa05202   hsa04360   hsa04062
## "hsa04310" "hsa04022" "hsa05152" "hsa05016" "hsa05202" "hsa04360" "hsa04062"
##   hsa04020   hsa04510   hsa05205   hsa04015   hsa04714   hsa04810   hsa04014
## "hsa04020" "hsa04510" "hsa05205" "hsa04015" "hsa04714" "hsa04810" "hsa04014"
##   hsa04060   hsa04010   hsa05165   hsa04151   hsa05200   hsa01100   hsa05206
## "hsa04060" "hsa04010" "hsa05165" "hsa04151" "hsa05200" "hsa01100" "hsa05206"
##   hsa05164
## "hsa05164"
viewGSEA(gsca3, gscName="GO_MF", gsName=topGS[["GO_MF"]][1])
```

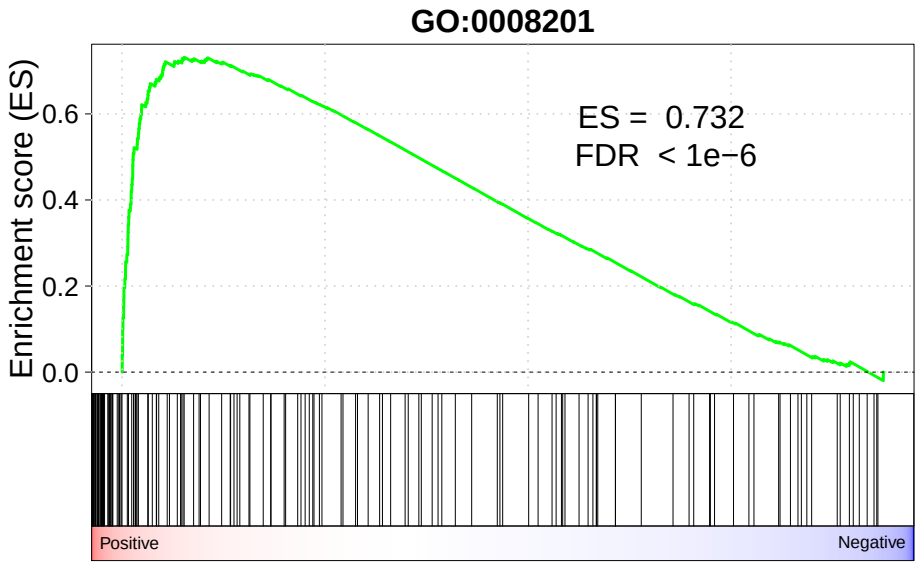

Figure 2: GSEA result plot of one gene set of the Molecular Function collection

You can also plot all or the top significant gene sets in batch and store them as png or pdf format into a specified path by using *plotGSEA*.

```
plotGSEA(gsca3, gscs=c("GO_MF", "PW_KEGG"), ntop=3, filepath=".")
```

2.2.6 Enrichment Map

To get a comprehensive view of the GSOA result or GSEA result instead of a list of significant gene sets with no relations, our package provides *viewEnrichMap* function to draw an enrichment map for better interpretation(Merico D (2010)). More specifically, in the enrichment map, nodes represent significant gene sets sized by the genes it contains and the edge represents the Jaccard similarity coefficient between two gene sets. Nodes color are scaled according to the adjusted pvalues (the darker, the more significant). For GSOA, there is only one color for nodes whereas for GSEA enrichment map, the default color is setted by the sign of enrichment scores (red:+, blue:-). You can also set your favourite format by changing the parameter named 'options'.

However, users are always highly recommended to use function *report* to visualize and modify the enrichment map with personal preference in an interactive report, such as different layout, color and size of nodes, types of labels and etc. More details please go to Part4: An interactive Shiny report of results.

```
## the enrichment map of GSEA result for top 5 significant
## gene sets in 'PW_KEGG' and 'GO_MF'
viewEnrichMap(gsca3, resultName = "GSEA.results",
              gscs=c("PW_KEGG", "GO_MF"),
              allSig = FALSE, gsNameType = "term", ntop = 8)
```

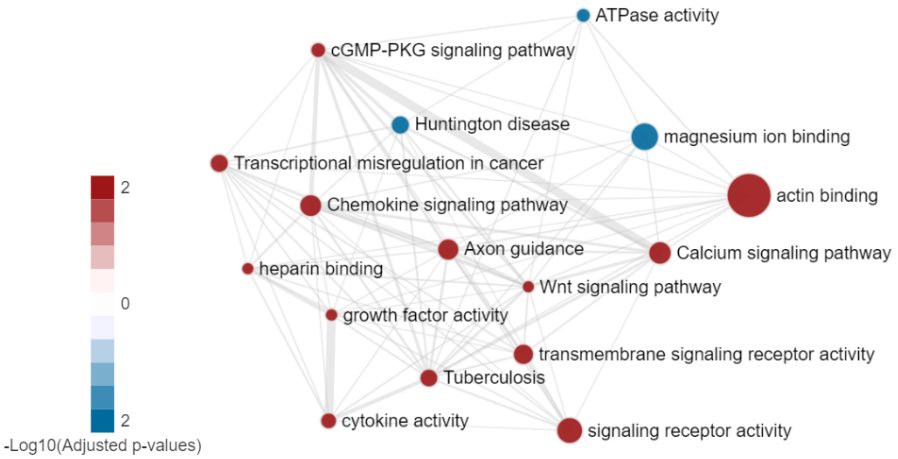

Figure 3: **Enrichment Map of GSEA result on top 8 significant gene sets of both "PW_KEGG" and "GO_MF"**

```
## the enrichment map of GSEA result for all significant
## gene sets in 'PW_KEGG'
viewEnrichMap(gsca3, resultName = "GSEA.results",
              gscs=c("PW_KEGG"),
              allSig = TRUE, gsNameType = "term")
```

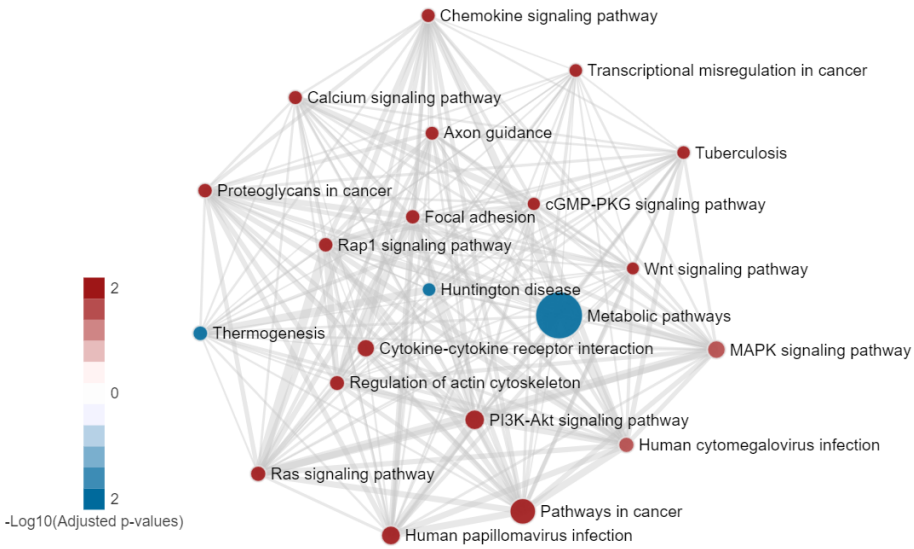

Figure 4: **Enrichment Map of GSEA result on all significant gene sets of "PW_KEGG"**

From the above GSEA enrichment maps, though the we filter many gene sets by a large cutoff to speed up the compilation of this vignette, we can still see that for the CMS4 CRC samples, pathways related to metabolism tend to be down-regulated and pathways related to cancer metastasis such as "Focal adhension" are significantly up-regulated, which is very consistent with what's been reported by Guinney J et al. in 2015 (Guinney J (2015))

```
## the enrichment map of GSOA result for
## top 10 significant gene sets in 'PW_KEGG' and 'GO_MF'
viewEnrichMap(gsca3, resultName = "HyperGeo.results",
              gscs=c("PW_KEGG", "GO_MF"),
              allSig = FALSE, gsNameType = "term", ntop = 10)
```

```
## the enrichment map of GSOA result for
## all significant gene sets in 'PW_KEGG'
viewEnrichMap(gsca3, resultName = "HyperGeo.results",
              gscs=c("PW_KEGG"),
              allSig = TRUE, gsNameType = "term")
```

From the above GSOA enrichment maps, the trend is quite similar as in GSEA result though here we have no idea of how these pathways are regulated.

2.2.7 Enrichment Map with specific gene sets

It is often the case that the enrichment map would be of large size due to a huge number of enriched gene sets. However, you may only be interested in a small part of them. A big size of enrichment map would also be in a mess and lose the information it can offer. In that way, **HTSanalyzeR2** provides an interface allowing users to draw the enrichment map on their interested gene sets. More details please see the help documentation of function *viewEnrichMap*.

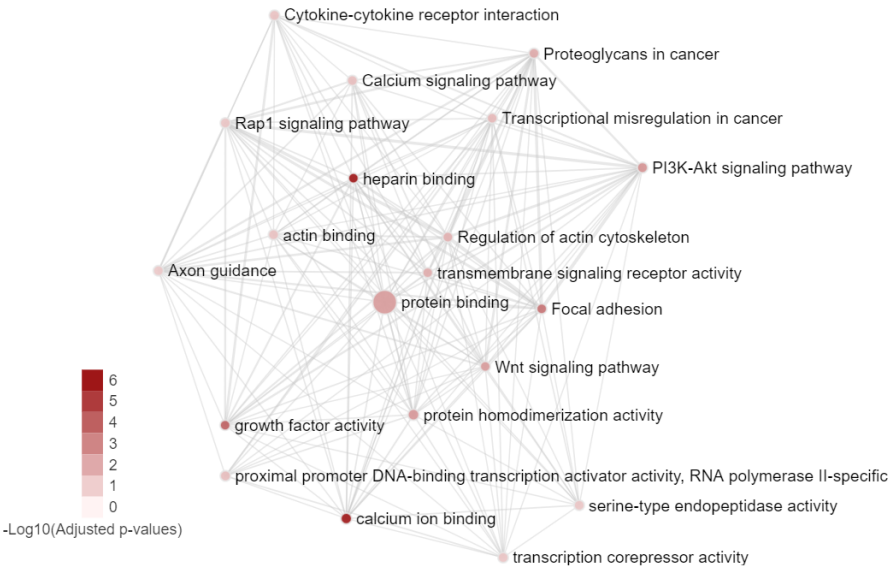

Figure 5: **Enrichment Map of GSOA result on top 10 significant gene sets of both "PW_KEGG" and "GO_MF"**

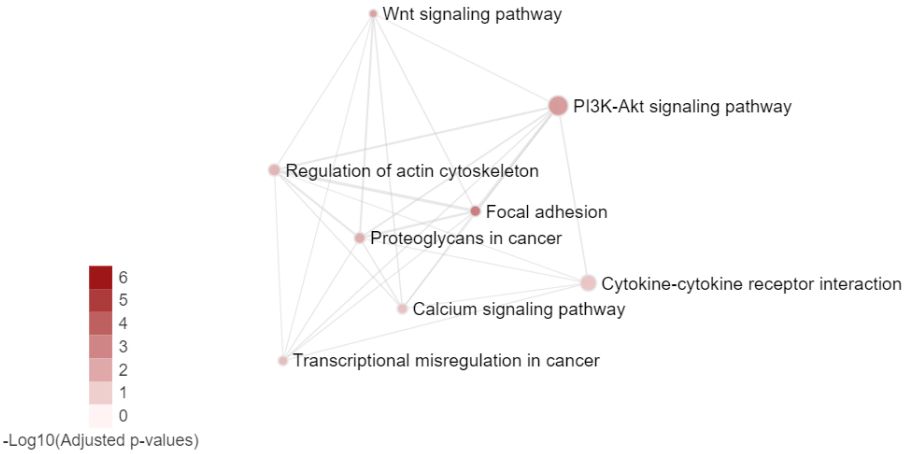

Figure 6: **Enrichment Map of GSOA result on all significant gene sets of "PW_KEGG"**

```
## specificGeneset needs to be a subset of all analyzed gene sets
## which can be roughly gotten by:
tmp <- getTopGeneSets(gsca3, resultName = "GSEA.results", gscs=c("PW_KEGG"),
                      ntop = 200, allSig = FALSE)

## In that case, we can define specificGeneset such as below:
PW_KEGG_geneset <- tmp$PW_KEGG[c(2, 3, 6, 7, 10, 18)]

## the name of specificGenesets also needs to match with the names of tmp
specificGeneset <- list("PW_KEGG"=PW_KEGG_geneset)
```

```
viewEnrichMap(gsca3, resultName = "GSEA.results", gscs=c("PW_KEGG"),
              allSig = FALSE, gsNameType = "term",
              ntop = NULL, specificGeneset = specificGeneset)
```

2.3 Enriched subnetwork analysis

You can also perform subnetwork analysis (Beisser (2010), Dittrich MT (2008)) to extract the subnetwork enriched with nodes which are associated with a significant phenotype using **HTSanalyzeR2**. The network can either be fetched by our package to download specific species network from BioGRID database or defined by users.

2.3.1 Prepare input, initialize and preprocess

An S4 class named 'NWA' is developed to perform subnetwork analysis. To initiate an 'NWA' object, you need to prepare a named numeric vector called 'pvalues'. If phenotypes for genes are also available, they can be inputted in the initialization step and used to highlight nodes with different colors in the identified subnetwork. In that case, the nodes are colored by the sign of phenotypes (red:+, blue:-).

When creating a new object of class 'NWA', the user also has the possibility to specify the parameter 'interactome' which should be an object of class 'igraph'. If it is not available, the interactome can also be set up later.

```
pvalues <- GSE33113_limma$adj.P.Val
names(pvalues) <- rownames(GSE33113_limma)
nwa <- NWA(pvalues=pvalues, phenotypes=phenotype)
```

The next step is to preprocess the inputs. Similar to 'GSCA' class, the function *preprocess* can conduct invalid input data removing, duplication removing by different methods and initial gene identifiers converting to Entrez ID.

```
nwa1 <- preprocess(nwa, species="Hs", initialIDs="SYMBOL",
                   keepMultipleMappings=TRUE, duplicateRemoverMethod="max")
```

Then, you need to create an interactome for the network analysis using method *interactome* if you have not inputted your own interactome in the initial step. To this end, you can either specify the species and fetch the corresponding network from BioGRID database, or input an interaction matrix if it is in right format: a matrix with a row for each interaction, and at least contains the three columns "InteractorA", "InteractorB" and "InteractionType", where the interactors are specified by Entrez ID. For more details please see *help(interactome)*.

Here, we just use *interactome* to download an interactome from BioGRID, which would meet user's requirements in most cases.

```
nwa2 <- interactome(nwa1, species="Hs", genetic=FALSE)

getInteractome(nwa2)
```

2.3.2 Perform analysis and view the identified subnetwork

Having preprocessed the input data and created the interactome, the subnetwork analysis could be performed by using the *analyze* method. This function will plot a figure showing the fitting of the BioNet model to your distribution of pvalues (Beisser (2010)), which is a good way to check the choice of statistics used in this function. The argument *fdr* of the method *analyze* is the false discovery rate for BioNet to fit the beta-uniform mixture (BUM) model. The parameters of the fitted model will then be used for the scoring function, which subsequently enables the BioNet package to search the optimal scoring subnetwork. See BioNet for more details (Beisser (2010)).

Here, to simplify this vignette, we set a very strict 'fdr' as 1e-06. In practice, you may want to set a less strict one (e.g. 0.01).

```
nwa3 <- analyze(nwa2, fdr=1e-06, species="Hs")
```

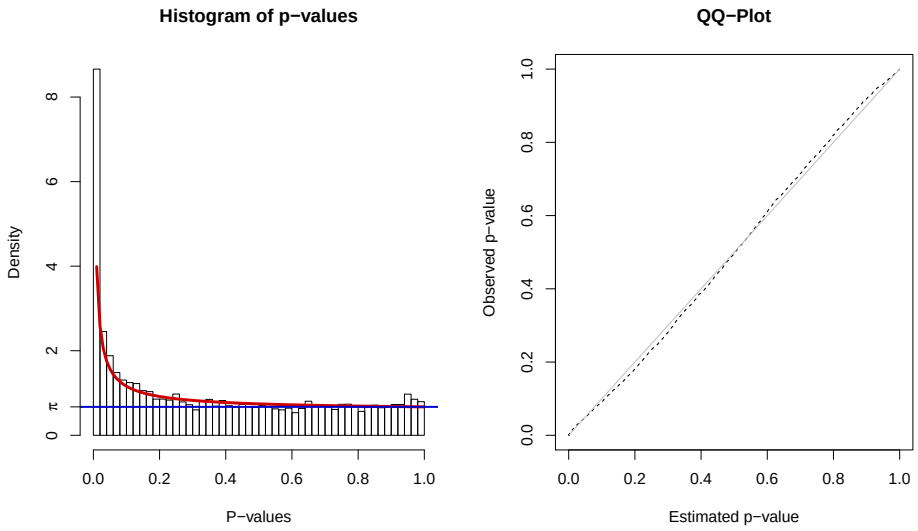

Figure 7: Fitting BUM model to p-values by BioNet

Similar to 'GSCA', you can also view the subnetwork by *viewSubNet*. Again, for better visualization, modification and downloading, users are highly recommended to view the result in an interactive Shiny report by function *report*.

```
viewSubNet(nwa3)
```

2.3.3 Summarize results

Like 'GSCA', the method *summarize* could also be used to get a general summary of an analyzed 'NWA' object including inputs, interactome, parameters for analysis and the size of identified subnetwork.

```
summarize(nwa3)
##
## -p-values:
##               input               valid   duplicate removed
```

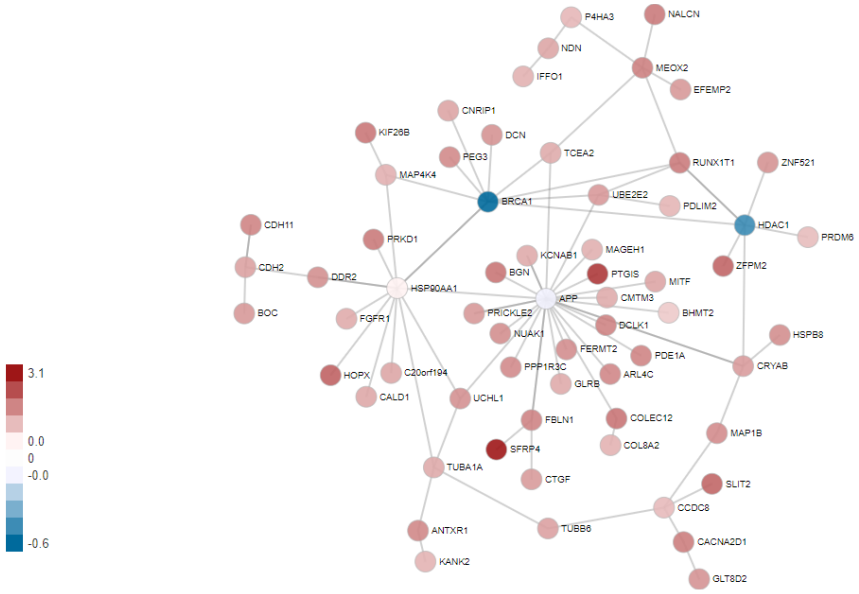

Figure 8: Enriched subnetwork identified by BioNet

```
##               21656               21655               21655
## converted to entrez      in interactome
##               18757               14958
##
##
## -Phenotypes:
##               input               valid   duplicate removed
##               21656               21655               21655
## converted to entrez      in interactome
##               18757               14958
##
##
## -Interactome:
##                     name    species genetic node No edge No
## Interaction dataset Biogrid Hs      FALSE   22439   332134
##
##
## -Parameters for analysis:
##             FDR
## Parameter 1e-06
##
##
## -Subnetwork identified:
##            node No edge No
## Subnetwork      81     113
```

3 Case study2: Time series analysis with CRISPR data

This case study uses a time series CRISPR genome-wide drop-out data as a demonstration to perform time series analysis. Data 'd7', 'd13' and 'd25' are three gRNA sequencing data after transducting the CRISPR system into a human cancer cellline on day 7, day 13 and day 25 separately (Tzelepis K (2016)), they are further preprocessed by MAGeCK to identify significant essential genes from genome-scale CRISPR knockout screens. Again, here to simply the compilation of this vignette, we skip the preprocessed steps and start from the results gotten by MAGeCK.

3.1 Gene set over-representation analysis (GSOA) and gene set enrichment analysis (GSEA)

3.1.1 Prepare the input data

To perform analysis for time series data, one must prepare the following inputs:

1. A character matrix contains experiment information with each experiment in row and descriptions in column. Specifically, it should at least contain two columns named as 'ID' and 'Description'.
2. A list of phenotypes, each element of this list is a phenotype of that experiment. **An important thing here needs to be noted is the order of each element of this list must match the order of 'ID' in the experiment information matrix**.
3. A list of gene set collections which can either be gotten by **HTSanalyzeR2** or defined by users using customized gene sets.

To make it easy to compile this vignette, here we only use Biological Process (BP) in Gene Ontology to make a demonstration.

```
data(d7, d13, d25)
expInfor <- matrix(c("d7", "d13", "d25"), nrow = 3, ncol = 2, byrow = FALSE,
                   dimnames = list(NULL, c("ID", "Description")))
datalist <- list(d7, d13, d25)

phenotypeTS <- lapply(datalist, function(x) {
  tmp <- as.vector(x$neg.lfc)
  names(tmp) <- x$id
  tmp
})

GO_BP <- GOGeneSets(species="Hs", ontologies=c("BP"))
ListGSC <- list(GO_BP=GO_BP)
```

Similar as single dataset analysis, if you also want to do GSOA, a list of hits is needed. Here, each element of this list is a hits of that experiment. Also, **the order of each element of this list must match the order of 'ID' in the experiment information matrix**. Here, for each data set, we define genes with *pvalue* less than 0.01 as hits.

```
hitsTS <- lapply(datalist, function(x){
  tmp <- x[x$neg.p.value < 0.01, "id"]
  tmp
})
```

3.1.2 Initialize and preprocess

To perform gene set enrichment analysis and hypermetric test for time-series data, an S4 class 'GSCABatch' which can pack the time series data to do further analysis is developed. First, you need to create a new 'GSCABatch' object using the prepared inputs.

```
gscaTS <- GSCABatch(expInfor = expInfor,
              phenotypeTS = phenotypeTS, listOfGeneSetCollections = ListGSC,
              hitsTS = hitsTS)
```

Then, the 'GSCABatch' object need to be preprocessed using *preprocessGscaTS* method. The preprocess procedure here is the same as single data set. This step would return a list of preprocessed 'GSCA' object.

```
gscaTS1 <- preprocessGscaTS(gscaTS, species="Hs", initialIDs="SYMBOL",
                            keepMultipleMappings=TRUE,
                            duplicateRemoverMethod="max",
                            orderAbsValue=FALSE)
```

3.1.3 Perform analysis

After getting a list of preprocessed 'GSCA' object, you can use *analyzeGscaTS* to perform GSOA as well as GSEA on it. The parameters' function here is the same as in single data set. Similarly, to speed up you can use multiple cores via *doParallel* package. This step would return a list of analyzed 'GSCA' object.

```
## analyze using 4 cores
if (requireNamespace("doParallel", quietly=TRUE)) {
    doParallel::registerDoParallel(cores=4)
} else {
}

gscaTS2 <- analyzeGscaTS(gscaTS1, para=list(pValueCutoff=0.05,
                                            pAdjustMethod="BH",
                                            nPermutations=100,
                                            minGeneSetSize=100,
                                            exponent=1),
                         doGSOA = TRUE, doGSEA = TRUE)

## GSEA result of the first experiment usingBP gene sets
head(getResult(gscaTS2[[1]])$GSEA.results$GO_BP, 3)
```

To make the result more understandable, users are highly recommended to annotate the gene sets ID to names by function *appendGSTermsTS*. As a result, an additional column named 'Gene.Set.Term' would appear.

```
gscaTS3 <- appendGSTermsTS(gscaTS2, goGSCs=c("GO_BP"))
head(getResult(gscaTS3[[1]])$GSEA.results$GO_BP, 3)
##                             Gene.Set.Term Observed.score Pvalue
## GO:0000398 mRNA splicing, via spliceosome     -0.8765546      0
## GO:0016579       protein deubiquitination     -0.7308489      0
## GO:0007049                     cell cycle     -0.6236592      0
##            Adjusted.Pvalue
## GO:0000398               0
## GO:0016579               0
## GO:0007049               0
```

You can then use *reportAll* to generate an interactive Shiny report to visualize a union enrichment map for this time series data. To put it more specific, a union enrichment map is generated by taking the union of significant gene sets in each experiment and then form an enrichment map as illustrated before. Thus there maybe be some gene sets not significant in one time point but in others. More details please see Part4: An interactive Shiny report of results.

3.2 Enriched subnetwork analysis

3.2.1 Prepare input, initialize and preprocess

An S4 class named 'NWABatch' is developed to pack time series data for further subnetwork analysis. You need first to create a new object of class 'NWABatch'. To this end, a list of pvalues is needed. Each element of this list is a vector of pvalues of that experiment. **Again, an important thing needs to be noted is the order of each element of this list must match the order of 'ID' in the experiment information matrix**. If a list of phenotypes is also available, they can be inputted during the initialization stage and used to highlight nodes with different colors in the identified subnetwork. Also, the order of each element of this phenotypes list must match the order of 'ID' in the experiment information matrix.

```
pvalueTS <- lapply(datalist, function(x){
  tmp <- as.vector(x$neg.p.value)
  names(tmp) <- x$id
  tmp
})

## generate a new 'NWABatch' object for further analysis
nwaTS <- NWABatch(expInfor = expInfor,
            pvalueTS = pvalueTS, phenotypeTS = phenotypeTS)
```

After creating an object of 'NWABatch', a preprocessing step needs to be performed which will return a list of preprocessed 'NWA' objects.

```
nwaTS1 <- preprocessNwaTS(nwaTS, species="Hs", initialIDs="SYMBOL",
                          keepMultipleMappings=TRUE,
                          duplicateRemoverMethod="max")
```

3.2.2 Perform analysis

Similarly, an interactome needs to be created before performing subnetwork analysis using *interactomeNwaTS* if you have not inputted your own interactome in the initial step. You can either specify the species and fetch the corresponding network from BioGRID database, or input an interaction matrix if it is in right format. More details please see *help(interactomeNwaTS)*.

Then, *analyzeNwaTS* could perform the subnetwork analysis for a list of 'NWA' object, which would take a few minutes. Finally, this step would return a list of analyzed 'NWA' objects.

```
nwaTS2 <- interactomeNwaTS(nwaTS1, species="Hs",
                           reportDir="HTSanalyzerReport", genetic=FALSE)
nwaTS3 <- analyzeNwaTS(nwaTS2, fdr=0.0001, species="Hs")
```

```
## get a general summary for the first experiment
summarize(nwaTS3[[1]])
```

You can then use *reportAll* to generate an interactive Shiny report to visualize a union subnetwork for this time series data. To put it more specific, a union subnetwork is generated by taking the union of identified subnetwork in each experiment. Thus there maybe be some genes not identified in the subnetwork of one time point but in others. More details please see Part4: An interactive Shiny report of results.

4 An interactive Shiny report

To better visualize all the results, our package could generate an interactive Shiny report containing all the results in. For single data set result such as 'gsca3' and 'nwa3' generated by the above analysis, you can either use *report* or *reportAll* as below:

```
## 1. generate an interactive shiny report for gsca object using report
report(gsca=gsca3)

## 2. generate an interactive shiny report for nwa object using report
report(nwa=nwa3)

## 3. generate an interactive shiny report for gsca object using reportAll
reportAll(gsca=gsca3)
```

You can even visualize both 'GSCA' and 'NWA' object with *reportAll* function. Particularly, to better visualize the enrichmentmap, you can filter away edges with small Jaccard coefficient by setting parameter 'cutoff'.

```
reportAll(gsca=gsca3, nwa=nwa3, cutoff = 0.08)
```

For time series data, you should use *reportAll* to generate the report. In addition, you can reset the order of time series data for visualization by setting the argument 'TSOrder'.

```
## 1. generate an interactive shiny report for
##    a list of gsca objects using reportAll
reportAll(gsca=gscaTS3)
```

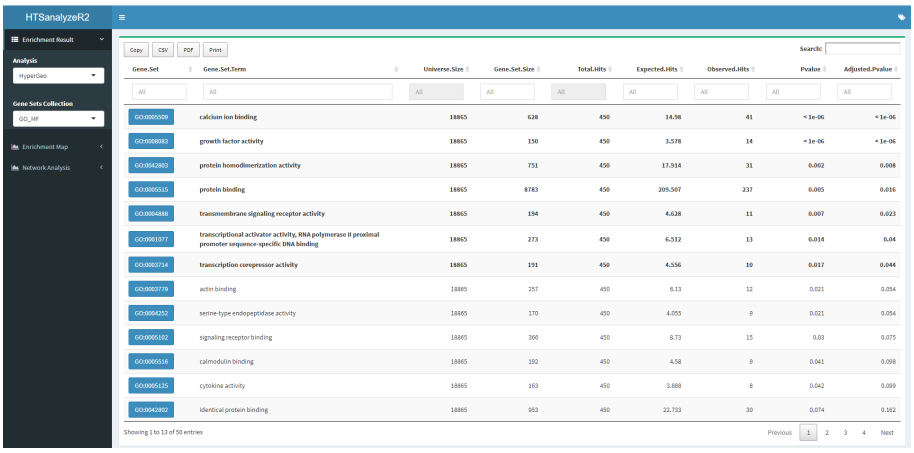

Figure 9: A screenshot of Shiny report on GSOA result table of 'gsca3' and 'nwa3'

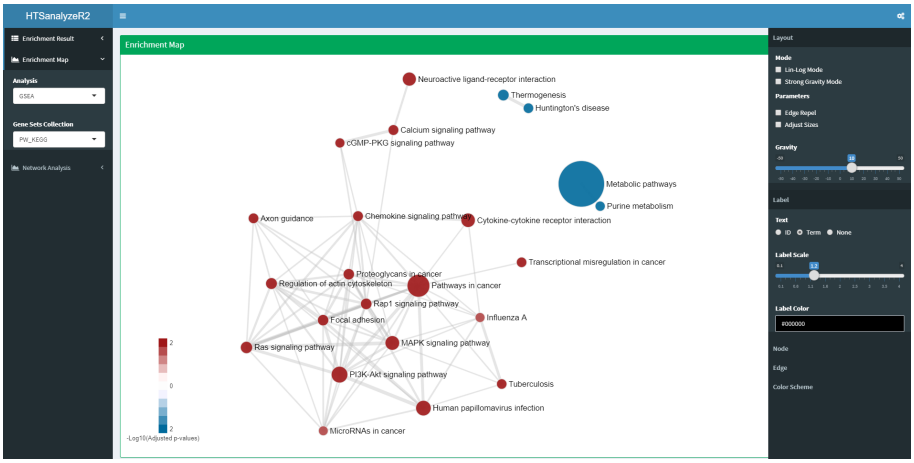

Figure 10: A screenshot of Shiny report on GSEA enrichment map of 'gsca3' and 'nwa3'

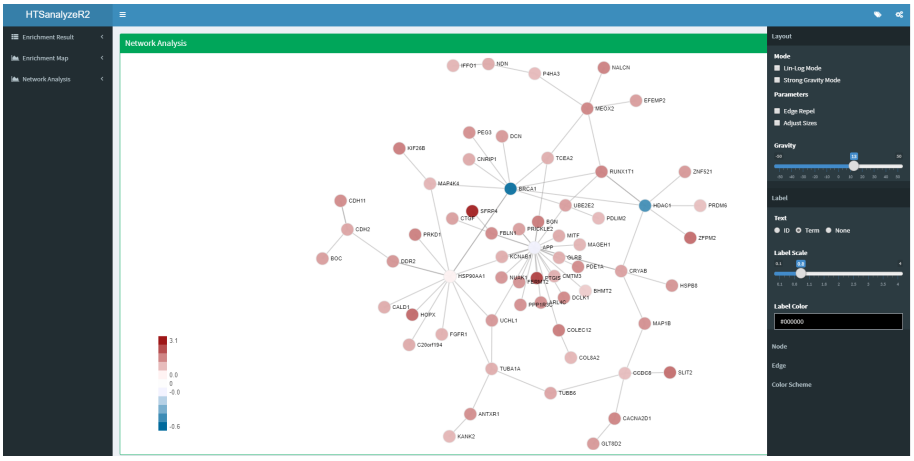

Figure 11: A screenshot of Shiny report on subnetwork of 'gsca3' and 'nwa3'

```
## 2. generate an interactive shiny report for
##    a list of nwa objects using reportAll
reportAll(nwa=nwaTS3)

## 3. generate an interactive shiny report for a list of
##    gsca objects by customized order using reportAll
reportAll(gsca=gscaTS3, TSOrder=names(gscaTS3)[c(3, 1, 2)])
```

In the interactive report, for GSOA and GSEA results of single data set, you can download the table of different gene set collection in different format such as 'csv' or 'pdf'. On the right of this interface, there is the summary information about this analysis. For the dynamic enrichment map, you can change the layout, set node size and color, label types, edge thickness and download it as 'svg' format. For subnetwork analysis result, you can also change the above mentioned items to fit your requirements.

Intriguingly, for time series data result, you can see a dynamic change for each 'time point' in either the union enrichment map or the union subnetwork, which could give you a general view about the difference among each tome point.

Similar with single data set analysis, you can also see the enrichment map of specific genesets for time series data by specify the argument 'specificGeneset' in *reportAll*.

```
## As told previously, specificGeneset needs to be a subset of all
## analyzed gene sets which can be roughly gotten by:
tmp <- getTopGeneSets(gscaTS3[[1]], resultName = "GSEA.results",
                      gscs=c("GO_BP"), ntop = 20000, allSig = FALSE)

## In that case, we can define specificGeneset as below:
GO_BP_geneset <- tmp$GO_BP[c(4,2,6,9,12, 15:20)]

## the name of specificGenesets also needs to match with the names of tmp
specificGeneset <- list("GO_BP"=GO_BP_geneset)
reportAll(gsca=gscaTS3, specificGeneset=specificGeneset, cutoff = 0.06)
```

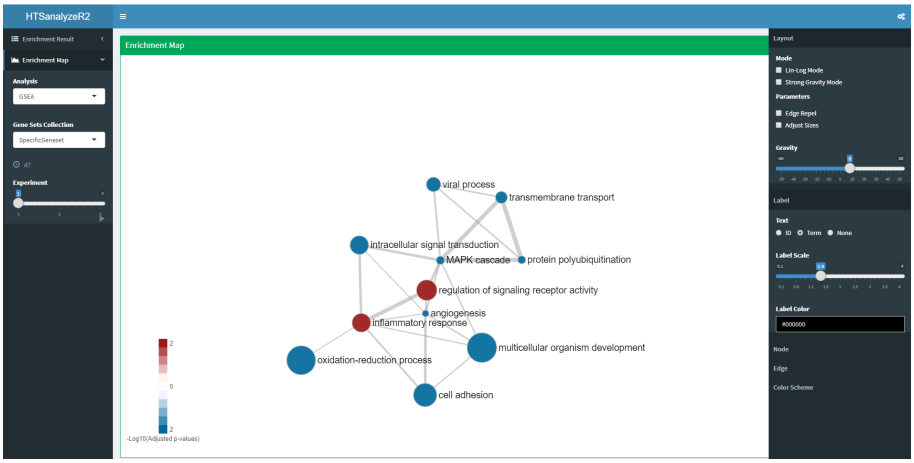

Figure 12: A screenshot of Shiny report on GSEA enrichment map with specific gene sets of 'gscaTS3'

After calling *report* or *reportAll*, it would automaticlly generate a directory with names starting with "GSCAReport", "NWAReport" or "AnalysisReport" which includes the result named as "results.RData" and an R script named as "app.R". Open "app.R" in RStudio, users can publish and share the report with others via Shinyapps.io, details please go to Shinyapps.io.

5 Special usage of HTSanalyzeR2

5.1 Gene set over-representation analysis (GSOA) with no background

In case if you only have a list of genes and want to do GSOA to measure the significance of overlapping with gene sets having known functions, **HTSanalyzeR2** provides an interface to realize it. Since phenotype is only used as background genes in GSOA, you can manually set all the genes of that species as phenotype and give them a pseudo value to fit **HTSanalyzeR2** as below:

```
data(d7)
hits1 <- d7$id[1:200]

## set all the coding genes of Homo sapiens as phenotype
allgenes <- keys(TxDb.Hsapiens.UCSC.hg19.knownGene, keytype = "GENEID")
## change Entrez ID to gene name to keep consistent with hits
allgenes <- mapIds(org.Hs.eg.db, keys = allgenes,
                   keytype = "ENTREZID", column = "SYMBOL")

## give phenotype a pseudo value to fit for HTSanalyzeR2
phenotype <- rep(1, length(allgenes))
names(phenotype) <- allgenes
```

Then, you can use the artificial phenotype as gene background and your hits to perform GSOA.

```
gsca <- GSCA(listOfGeneSetCollections=ListGSC,
            geneList=phenotype, hits=hits1)
## the following analysis is the same as before
```

5.2 Customized gene sets

When you have your own gene sets with specific functions and they do not belong to any GO terms, KEGG or MSigDB. In that case, you can set your customized gene set collection and follow the format of GO, KEGG and MSigDB gene set collections. An important thing here you need pay attention to is the ID of genes in the gene set collection must be Entrez ID.

```
## Suppose your own gene sets is geneset1 and geneset2
allgenes <- keys(org.Hs.eg.db, "ENTREZID")
geneset1 <- allgenes[sample(length(allgenes), 100)]
geneset2 <- allgenes[sample(length(allgenes), 60)]
```

```
## Set your custom gene set collection and make the format to fit HTSanalyzeR2
CustomGS <- list("geneset1" = geneset1, "geneset2" = geneset2)

## then the gene set collections would be as below:
ListGSC <- list(CustomGS=CustomGS)
## other part is the same as before
```

5.3 An interface to 'fgsea' package

HTSanalyzeR2 also provides an interface to fgsea package which proposes a novel algorithm for fast preranked gene set enrichment analysis using cumulative statistic calculation. Details please see fgsea (Sergushichev (2016)).

To perform GSEA by **fgsea** instead of HTSanlayzeR2, users need to specifiy the parameter *GSEA.by* in *analyze* (for single data set) or *analyzeGscaTS* (for time series analysis) with *fgsea*. Otherwise, it will use **HTSanalyzeR2** by default.

```
gsca4 <- analyze(gsca1,
                 para=list(pValueCutoff=0.05, pAdjustMethod="BH",
                           nPermutations=100, minGeneSetSize=150,
                           exponent=1),
                 doGSOA = TRUE, doGSEA = TRUE,
                 GSEA.by = "fgsea")
```

5.4 Extract shared genes between enriched pathways and input gene list

Once you've finished the GSEA or GSOA, you may be interested in some pathways and wonder which genes are shared by that pathway and you input gene list.

```
## Suppose you are interested in "growth factor activity" in 'Molecular Function',
## of Gene Ontology, We can retrieve the GO ID of this pathway:
GO_MFrslt <- getResult(gsca3)$HyperGeo.results$GO_MF
GOID <- rownames(GO_MFrslt[GO_MFrslt$Gene.Set.Term ==
                                          "growth factor activity", ])

## Then get the genes in this pathways
pathway_gene <- GO_MF[[GOID]]

## change Entrez ID to gene symbol
pathway_gene <- mapIds(org.Hs.eg.db, keys = pathway_gene,
                       keytype = "ENTREZID", column = "SYMBOL")
## 'select()' returned 1:1 mapping between keys and columns

## get the shared genes between this pathway and your input gene list
intersect(pathway_gene, hits)
##  [1] "CTGF"    "EFEMP1"  "FGF7"    "IGF1"    "INHBA"   "NOV"     "OGN"
##  [8] "CLEC11A" "CXCL12"  "TGFB3"   "THBS4"   "VEGFC"   "DKK1"    "PDGFC"
```

6 A pipeline function for common phenotype data

For common phenotype data, we provide a pipeline function *HTSanalyzeR2Pipe* to perform a comprehensive analysis including GSEA and subnetwork analysis. Finally, it will return a list of *GSCA* object and *NWA* object.

```
data4enrich <- as.vector(d7$neg.lfc)
names(data4enrich) <- d7$id
hits <-  names(data4enrich[which(abs(data4enrich) > 2)])
ListGSC = list(GO_MF=GO_MF, PW_KEGG=PW_KEGG)
rslt <- HTSanalyzeR2Pipe(data4enrich = data4enrich,
                         hits = hits,
                         doGSOA = TRUE,
                         doGSEA = TRUE,
                         listOfGeneSetCollections = ListGSC,
                         species = "Hs",
                         initialIDs = "SYMBOL",
                         pValueCutoff = 0.05,
                         nPermutations = 1000,
                         cores = 2,
                         minGeneSetSize = 15,
                         keggGSCs=c("PW_KEGG"),
                         goGSCs = c("GO_MF"),
                         doNWA = FALSE)
report(rslt$gsca)
```

7 A pipeline function for CRISPR data pre-processed by MAGeCK

For the CRISPR data pre-processed by MAGeCK, we also provide a pipeline function to do a comprehensive analysis including GSEA and subnetwork analysis, which would be seamless linking with MAGeCK and provides great convenience to the users. Finally, it would automaticlly generate a dynamic shiny report containing all the results.

```
ListGSC = list(GO_MF=GO_MF, PW_KEGG=PW_KEGG)
HTSanalyzeR4MAGeCK(MAGeCKdata = d7,
                   selectDirection = "negative",
                   doGSOA = FALSE,
                   doGSEA = TRUE,
                   listOfGeneSetCollections = ListGSC,
                   species = "Hs",
                   initialIDs = "SYMBOL",
                   pValueCutoff = 0.05,
                   pAdjustMethod = "BH",
                   nPermutations = 100,
                   minGeneSetSize = 100,
                   exponent = 1,
                   keggGSCs=c("PW_KEGG"),
```

```
                        goGSCs = c("GO_MF"),
                        msigdbGSCs = NULL,
                        doNWA = TRUE,
                        reportDir = "HTSanalyzerReport",
                        nwAnalysisGenetic = FALSE,
                        nwAnalysisFdr = 0.001)
```

8 Session Info

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.1 LTS
##
## Matrix products: default
## BLAS: /usr/lib/x86_64-linux-gnu/openblas/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/libopenblasp-r0.2.20.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [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
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2
##  [2] GenomicFeatures_1.34.3
##  [3] GenomicRanges_1.34.0
##  [4] GenomeInfoDb_1.18.1
##  [5] limma_3.38.3
##  [6] igraph_1.2.2
##  [7] GO.db_3.7.0
##  [8] KEGGREST_1.22.0
##  [9] org.Hs.eg.db_3.7.0
## [10] AnnotationDbi_1.44.0
## [11] IRanges_2.16.0
## [12] S4Vectors_0.20.1
## [13] Biobase_2.42.0
## [14] BiocGenerics_0.28.0
## [15] HTSanalyzeR2_0.99.15
## [16] BiocStyle_2.10.0
##
## loaded via a namespace (and not attached):
```

```
##   [1] fgsea_1.8.0                colorspace_1.4-0
##   [3] hwriter_1.3.2              XVector_0.22.0
##   [5] affyio_1.52.0              DT_0.5
##   [7] bit64_0.9-7                mvtnorm_1.0-8
##   [9] codetools_0.2-16           splines_3.5.2
##  [11] doParallel_1.0.14          robustbase_0.93-3
##  [13] knitr_1.21                 splots_1.48.0
##  [15] Rsamtools_1.34.1           prada_1.58.1
##  [17] annotate_1.60.0            cluster_2.0.7-1
##  [19] Rmpfr_0.7-2                vsn_3.50.0
##  [21] png_0.1-7                  shinydashboard_0.7.1
##  [23] graph_1.60.0               shiny_1.2.0
##  [25] BiocManager_1.30.4         rrcov_1.4-7
##  [27] compiler_3.5.2             httr_1.4.0
##  [29] assertthat_0.2.0           Matrix_1.2-15
##  [31] lazyeval_0.2.1             later_0.7.5
##  [33] prettyunits_1.0.2          htmltools_0.3.6
##  [35] tools_3.5.2                gmp_0.5-13.2
##  [37] bindrcpp_0.2.2             GenomeInfoDbData_1.2.0
##  [39] gtable_0.2.0               glue_1.3.0
##  [41] affy_1.60.0                Category_2.48.0
##  [43] dplyr_0.7.8                fastmatch_1.1-0
##  [45] Rcpp_1.0.0                 Biostrings_2.50.2
##  [47] preprocessCore_1.44.0      rtracklayer_1.42.1
##  [49] iterators_1.0.10           xfun_0.4
##  [51] stringr_1.3.1              mime_0.6
##  [53] miniUI_0.1.1.1             XML_3.98-1.17
##  [55] BioNet_1.42.0              DEoptimR_1.0-8
##  [57] zlibbioc_1.28.0            MASS_7.3-51.1
##  [59] scales_1.0.0               RankProd_3.8.0
##  [61] colourpicker_1.0           hms_0.4.2
##  [63] promises_1.0.1             SummarizedExperiment_1.12.0
##  [65] RBGL_1.58.1                RColorBrewer_1.1-2
##  [67] curl_3.3                   yaml_2.2.0
##  [69] memoise_1.1.0              gridExtra_2.3
##  [71] ggplot2_3.1.0              biomaRt_2.38.0
##  [73] stringi_1.2.4              RSQLite_2.1.1
##  [75] genefilter_1.64.0          cellHTS2_2.46.1
##  [77] pcaPP_1.9-73               foreach_1.4.4
##  [79] BiocParallel_1.16.5        matrixStats_0.54.0
##  [81] rlang_0.3.1                pkgconfig_2.0.2
##  [83] bitops_1.0-6               evaluate_0.12
##  [85] lattice_0.20-38            purrr_0.3.0
##  [87] bindr_0.1.1                GenomicAlignments_1.18.1
##  [89] htmlwidgets_1.3            bit_1.1-14
##  [91] tidyselect_0.2.5           GSEABase_1.44.0
##  [93] plyr_1.8.4                 magrittr_1.5
##  [95] bookdown_0.9               R6_2.3.0
##  [97] DelayedArray_0.8.0         DBI_1.0.0
##  [99] pillar_1.3.1               survival_2.43-3
## [101] RCurl_1.95-4.11            tibble_2.0.1
```

```
## [103] crayon_1.3.4              rmarkdown_1.11
## [105] progress_1.2.0            locfit_1.5-9.1
## [107] grid_3.5.2                data.table_1.12.0
## [109] blob_1.1.1                digest_0.6.18
## [111] xtable_1.8-3              httpuv_1.4.5.1
## [113] munsell_0.5.0
```

References

Arthur Liberzon, Reid Pinchback, Aravind Subramanian. 2011. "Molecular Signatures Database (Msigdb) 3.0." *Bioinformatics* 27 (12): 1739–40. doi:10.1093/bioinformatics/btr260.

Beisser, Klau, D. 2010. "BioNet: An R-Package for the Functional Analysis of Biological Networks." *Bioinformatics* 26 (8): 1129–30. doi:10.1093/bioinformatics/btq089.

Dittrich MT, Rosenwald A, Klau GW. 2008. "Identifying Functional Modules in Protein–Protein Interaction Networks: An Integrated Exact Approach." *Bioinformatics* 24 (13): i223–i231. doi:10.1093/bioinformatics/btn161.

Guinney J, Wang X, Dienstmann R. 2015. "The Consensus Molecular Subtypes of Colorectal Cancer." *Nature Medicine* 21 (11): 1350–6. doi:10.1038/nm.3967.

Merico D, Stueker O, Isserlin R. 2010. "Enrichment Map: A Network-Based Method for Gene-Set Enrichment Visualization and Interpretation." *PLoS ONE* 5 (11): e13984. doi:10.1371/journal.pone.0013984.

Sergushichev, Alexey. 2016. "An Algorithm for Fast Preranked Gene Set Enrichment Analysis Using Cumulative Statistic Calculation." *bioRxiv*. doi:10.1101/060012.

Subramanian A, Mootha VK, Tamayo P. 2005. "Gene Set Enrichment Analysis: A Knowledge-Based Approach for Interpreting Genome-Wide Expression Profiles." *Proceedings of the National Academy of Sciences of the United States of America* 102 (43): 15545–50. doi:10.1073/pnas.0506580102.

Tzelepis K, De Braekeleer E, Koike-Yusa H. 2016. "A Crispr Dropout Screen Identifies Genetic Vulnerabilities and Therapeutic Targets in Acute Myeloid Leukemia." *Cell Reports* 17 (4): 1193–1205. doi:10.1016/j.celrep.2016.09.079.