HTSanalyzeR2: An R package for gene set enrichment and network analysis of various 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

2018-03-27

Abstract

This package provides gene set over-representation, enrichment analysis and enriched subnetwork analyses for various preprocessed high-throughput data as well as corresponding time-series data generated by either CRISPR, 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 0.99.11

Contents

1 An overview of HTSanalyzeR2

While high-throughput experiments are no longer bottlenecks for biologists to dissect the functional mechanisms genome-widely, how to efficiently interpret and visualize the results remains a challenge. 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 over-representation, enrichment 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 a similar experiment coming from different groups, **HTSanalyzeR2** can perform either longitudinal or horizontal 'time-series' analysis for mutual comparing.
- **HTSanalyzeR2** could generate an interactive report for users downloading, modifying the visualizations as well as sharing with others.

1.1 Supported analysis

- Over-representation analysis
- Gene set enrichment analysis
- Enriched subnetwork analysis

1.2 Supported data types

- Named phenotypes prepressed from either **RNA-seq**, **micro-array**, **RNAi** and **CRISPR**
- **'Time Series' data**

1.3 Supported ontologies/pathways

- Gene Ontology:Molecular function(MF), Biological process(BP), Cellular component(CC)
- Kyoto Encyclopedia of Genes and Genomes pathways: KEGG
- Molecular Signatures Database: MSigDB v6.1
  - c2:curated gene sets
  - c5:GO gene sets
- Customized gene sets

1.4 Supported species

Gene Ontology and KEGG analysis support any species that have an **OrgDb** object in Bioconductor.

MSigDB analysis only supports **Homo Sapiens** because most gene sets in MSigDB are for Homo Sapiens.

1.5 Visualization

- gsea plots
- enrichment map
- enriched subnetwork
- interactive report for all results

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

```
library(HTSanalyzeR2)
library(org.Hs.eg.db)
library(KEGGREST)
library(GO.db)
library(igraph)
library(limma)
```

2 Case study1: Single dataset analysis for gene expression data

This case study is using **HTSanalyzeR2** to perform gene set over-representation, enrichment and network analyses on 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. However, to make this vignette simple enough, we skip to exlpain this step and start from the result gotten by 'limma'.

2.1 Hypergeometric test and gene set enrichment analysis

2.1.1 Prepare the input data

To perform gene set enrichment analysis for single dataset, you must preprare the following inputs:

1. a named numeric vector of phenotypes(normally this would be a vector of genes with log2 fold change).
2. a list of gene set collections.

First you need to prepare a named phenotype.

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

Then, if you also want to do hypergeometric test on a list of interested genes, 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. In this case, the names of phenotpe, 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 hypergeometric test with only a list of hits and no phenotype, **HTSanalyzeR2** can also realize it. For details please go to Part 5:Special usage of HTSanalyzeR2.

```
## define hits if you want to do hypergeometric test
hits <-  names(phenotype[which(abs(phenotype) > 1)])
```

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 a similar 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), KEGG pathways. Gene sets in a comprehensive molecular signatures database, MSigDB(Arthur Liberzon (2011)), for Homo Sapiens is also provided, containing the most commonly used two collections:'c2'(curated gene sets) and 'c5'(GO gene sets). 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:

```
GO_MF <- GOGeneSets(species="Hs", ontologies=c("MF"))
PW_KEGG <- KeggGeneSets(species="Hs")
MSig_C2 <- MSigDBGeneSets(collection = "c2")
ListGSC <- list(GO_MF=GO_MF, PW_KEGG=PW_KEGG, MSig_C2=MSig_C2)
```

2.1.2 Initialize and preprocess

An S4 class named 'GSCA' is developed to perform hypergeometric test in order to find the gene sets sharing significant overlapping with hits. Gene set enrichment analysis, 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 hypergeometric test, 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)
```

2.1.3 Perform analysis

After getting a preprocessed 'GSCA' object, you can perform hypergeometric test and gene set enrichment analysis using the function named *analyze*. This funciton 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 hypergeometric test 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=180,
                           exponent=1),
                 doGSOA = TRUE, doGSEA = TRUE)
```

In this case study, we only use 100 permutations and set a very 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 permutations(e.g. 1000) to get a more robust 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 perform 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 2 cores
if (requireNamespace("doParallel", quietly=TRUE)) {
    doParallel::registerDoParallel(cores=2)
} else {
}

gsca2 <- analyze(gsca1,
                 para=list(pValueCutoff=0.05, pAdjustMethod="BH",
                           nPermutations=100, minGeneSetSize=180,
                           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 hypergeometric test and GSEA are both performed, gene sets which are both significant in this two kinds of analysis based on either pvalue or adjusted pvalue can be accessed.

```
head(getResult(gsca2)$HyperGeo.results$GO_MF, 3)
##            Universe Size Gene Set Size Total Hits Expected Hits
## GO:0005509         18978           617        477     15.507904
## GO:0042803         18978           720        477     18.096744
```

```
## GO:0003714         18978          184        477      4.624723
##             Observed Hits       Pvalue Adjusted.Pvalue
## GO:0005509            43 1.685852e-09    1.263131e-08
## GO:0042803            31 2.641243e-03    8.780821e-03
## GO:0003714            10 1.820557e-02    4.835554e-02
head(getResult(gsca2)$GSEA.results$PW_KEGG, 3)
##           Observed.score Pvalue Adjusted.Pvalue
## hsa05016      -0.5195224      0               0
## hsa04510       0.6267835      0               0
## hsa05205       0.5519429      0               0
head(getResult(gsca2)$Sig.pvals.in.both$MSig_C2, 3)
##                                                    HyperGeo.Pvalue
## TURASHVILI_BREAST_DUCTAL_CARCINOMA_VS_DUCTAL_NORMAL_DN    1.984818e-17
## TONKS_TARGETS_OF_RUNX1_RUNX1T1_FUSION_HSC_UP              1.471194e-07
## MOHANKUMAR_TLX1_TARGETS_DN                                1.316622e-05
##                                                    GSEA.Pvalue
## TURASHVILI_BREAST_DUCTAL_CARCINOMA_VS_DUCTAL_NORMAL_DN           0
## TONKS_TARGETS_OF_RUNX1_RUNX1T1_FUSION_HSC_UP                     0
## MOHANKUMAR_TLX1_TARGETS_DN                                       0
head(getResult(gsca2)$Sig.adj.pvals.in.both$MSig_C2, 3)
##                                                    HyperGeo.Adj.Pvalue
## TURASHVILI_BREAST_DUCTAL_CARCINOMA_VS_DUCTAL_NORMAL_DN        2.930525e-16
## TONKS_TARGETS_OF_RUNX1_RUNX1T1_FUSION_HSC_UP                  9.117773e-07
## MOHANKUMAR_TLX1_TARGETS_DN                                    6.884838e-05
##                                                    GSEA.Adj.Pvalue
## TURASHVILI_BREAST_DUCTAL_CARCINOMA_VS_DUCTAL_NORMAL_DN               0
## TONKS_TARGETS_OF_RUNX1_RUNX1T1_FUSION_HSC_UP                         0
## MOHANKUMAR_TLX1_TARGETS_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
## hsa05016    Huntington's disease     -0.5195224      0               0
## hsa04510          Focal adhesion      0.6267835      0               0
## hsa05205 Proteoglycans in cancer      0.5519429      0               0
```

2.1.4 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)
##
```

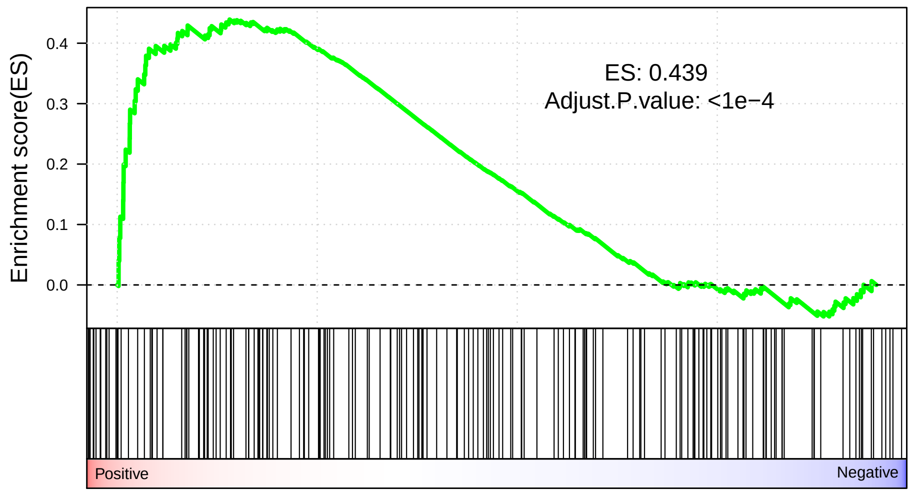
```
## -No of genes in Gene set collections:
##         input above min size
## GO_MF    4110             41
## PW_KEGG   327             20
## MSig_C2  3762            441
##
##
## -No of genes in Gene List:
##           input valid duplicate removed converted to entrez
## Gene List 21656 21655             21655               18978
##
##
## -No of hits:
##      input preprocessed
## Hits   496          477
##
##
## -Parameters for analysis:
##               minGeneSetSize pValueCutoff pAdjustMethod
## HyperGeo Test 180            0.05         BH
##
##      minGeneSetSize pValueCutoff pAdjustMethod nPermutations exponent
## GSEA 180            0.05         BH            100           1
##
##
## -Significant gene sets (adjusted p-value< 0.05 ):
##          GO_MF PW_KEGG MSig_C2
## HyperGeo     3       4     185
## GSEA        18      14     341
## Both         3       4     170
```

2.1.5 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:0003714   GO:0004872   GO:0004871   GO:0005096   GO:0003779
## "GO:0003714" "GO:0004872" "GO:0004871" "GO:0005096" "GO:0003779"
##   GO:0001077   GO:0004930   GO:0005102   GO:0003682   GO:0005509
## "GO:0001077" "GO:0004930" "GO:0005102" "GO:0003682" "GO:0005509"
##   GO:0042803   GO:0003723   GO:0003677   GO:0005524   GO:0005515
```

```
## "GO:0042803" "GO:0003723" "GO:0003677" "GO:0005524" "GO:0005515"
##   GO:0003924   GO:0045296   GO:0019899
## "GO:0003924" "GO:0045296" "GO:0019899"
##
## $PW_KEGG
##   hsa05016   hsa04510   hsa05205   hsa04015   hsa04810   hsa04714   hsa04014
## "hsa05016" "hsa04510" "hsa05205" "hsa04015" "hsa04810" "hsa04714" "hsa04014"
##   hsa04060   hsa04010   hsa05165   hsa04151   hsa05200   hsa01100   hsa04080
## "hsa04060" "hsa04010" "hsa05165" "hsa04151" "hsa05200" "hsa01100" "hsa04080"
viewGSEA(gsca3, gscName="GO_MF", gsName=topGS[["GO_MF"]][1])
```



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=".")
```

2.1.6 Enrichment Map

To get a comprehensive view of the hypergeometric test 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 hypergeometric test, 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 for top 5 significant gene sets in 'PW_KEGG' and 'GO_MF'
viewEnrichMap(gsca3, gscs=c("PW_KEGG", "GO_MF"),
```

```
              allSig = FALSE, gsNameType = "term", ntop = 5)
```

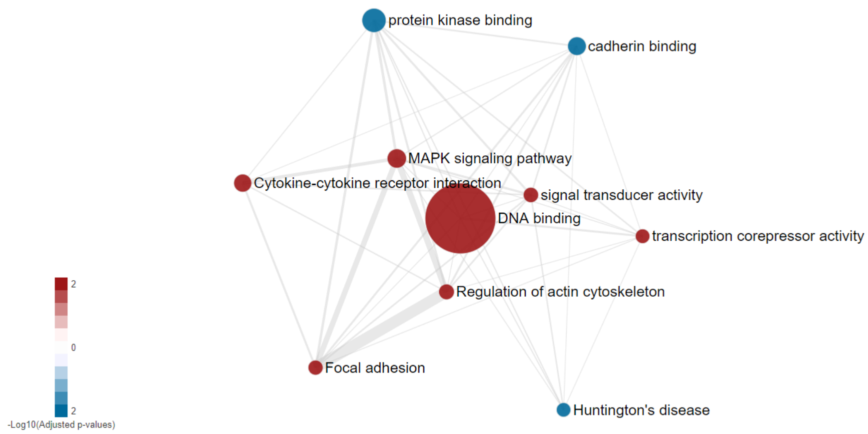


Figure 1:

2.1.7 Enrichment Map with specific gene sets

It is often the case that the enrichment map would be of large size due to the 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*.

```
## 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 = 20000, allSig = FALSE)
## In that case, we can define specificGeneset such as below:
PW_KEGG_geneset <- tmp$PW_KEGG[c(2, 3, 6, 7, 10, 11)]
## 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)
```

2.2 Enriched subnetwork analysis

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

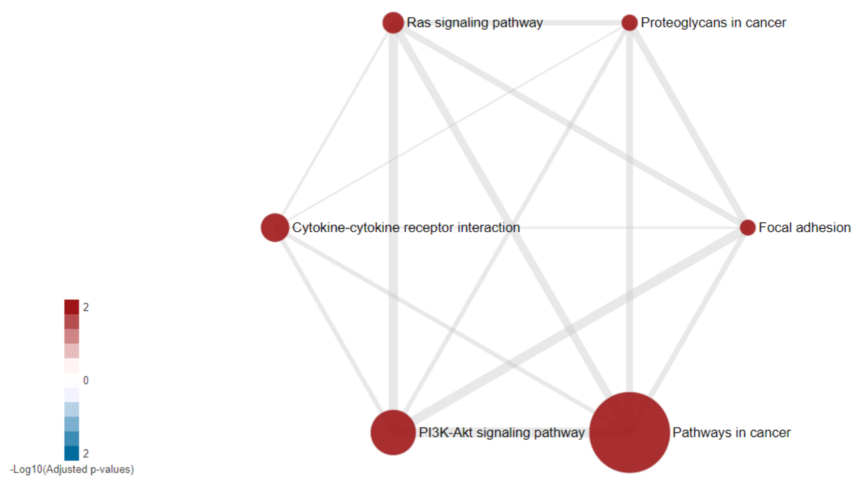


Figure 2:

2.2.1 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)
```

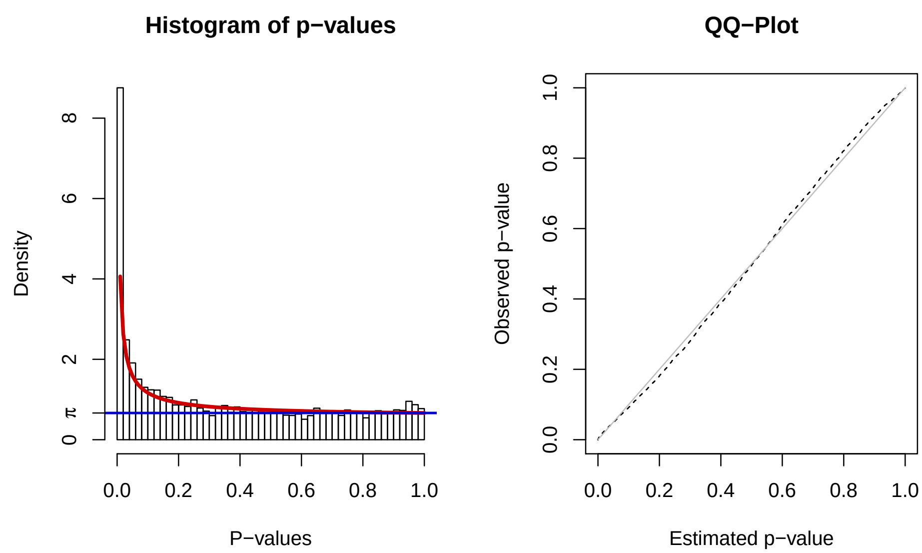
```
getInteractome(nwa2)
## IGRAPH 67d5fda UN-- 20223 258975 --
## + attr: name (v/c)
## + edges from 67d5fda (vertex names):
##  [1] 6416 --2318  84665--88    90   --2339  2624 --5371  6118 --6774
##  [6] 375  --23163 377  --23647 377  --27236 54464--226   351  --10513
## [11] 333  --1600  10370--7020  7020 --2033  338  --4547  409  --5900
## [16] 1436 --2885  2885 --7916  27257--4677  6521 --22950 602  --580
## [21] 153  --10755 672  --466   672  --4436  580  --672   672  --2956
## [26] 421  --1013  5092 --775   5664 --823   825  --7273  3708 --767
## [31] 9223 --1499  5925 --1523  7251 --1026  4998 --4171  4171 --5000
## [36] 4171 --4174  4171 --8317  4171 --4999  6118 --4171  4171 --10926
## + ... omitted several edges
```

2.2.2 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-07. In practice, you may want to set a less strict one(e.g. 0.01)

```
nwa3 <- analyze(nwa2, fdr=1e-07, species="Hs")
```



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)
```

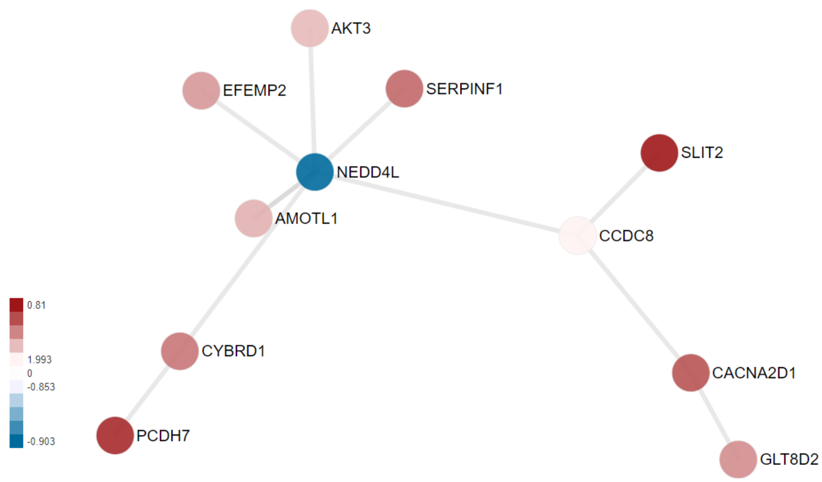


Figure 3:

2.2.3 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
##                21656              21655             21655
## converted to entrez      in interactome
##                18978              14346
##
##
## -Phenotypes:
##                input              valid   duplicate removed
##                21656              21655             21655
## converted to entrez      in interactome
##                18978              14346
##
##
## -Interactome:
##                      name    species genetic node No edge No
## Interaction dataset Biogrid Hs      FALSE   20223   258975
##
##
```

```
## -Parameters for analysis:
##             FDR
## Parameter 1e-07
##
##
## -Subnetwork identified:
##            node No edge No
## Subnetwork      11      11
```

3 Case study2: Time series analysis for 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 as time goes by(Tzelepis K (2016)), they are further preprocessed by MAGeCK to identify significant essential genes from genome-scale CRISPR knockout screens.

3.1 Hypergeometric test and gene set enrichment analysis

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 phenptypes, 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 be gotten by our package.

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 hypergeometric test, 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 get a list of preprocessed 'GSCA' object, you can use *analyzeGscaTS* to perform hypergeometric test 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 2 cores
if (requireNamespace("doParallel", quietly=TRUE)) {
    doParallel::registerDoParallel(cores=2)
} else {
}

gscaTS2 <- analyzeGscaTS(gscaTS1, para=list(pValueCutoff=0.05,
                                            pAdjustMethod="BH",
                                            nPermutations=100,
                                            minGeneSetSize=100,
                                            exponent=1),
                         doGSOA = TRUE, doGSEA = TRUE)
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:0007268 chemical synaptic transmission     -0.2125714      0
## GO:0000398 mRNA splicing, via spliceosome     -0.8695551      0
## GO:0000165                   MAPK cascade     -0.7162140      0
##            Adjusted.Pvalue
## GO:0007268               0
## GO:0000398               0
## GO:0000165               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 a time point. 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
})
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")
```

```
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 a time point. More details please see Part4:An interactive Shiny report of results.

4 An interactive Shiny report of results

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:

```
report(gsca=gsca3)
report(nwa=nwa3)
reportAll(gsca=gsca3)
reportAll(gsca=gsca3, nwa=nwa3)
```

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

```
reportAll(gsca=gscaTS3)
reportAll(nwa=nwaTS3)
reportAll(gsca=gscaTS3, TSOrder=names(gscaTS3)[c(3, 1, 2)])
```

In the interactive report, for hypergeometric test 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 direct view about the difference.

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)]
## 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)
```

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 Hypergeometric test with no phenotype

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

```
data(d7)
hits <- d7$id[1:200]
## set all the genes of Homo sapiens as phenotype
allgenes <- keys(org.Hs.eg.db, keytype = "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 and your hits to perform hypergeometric test.

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

5.2 Customized gene set

When you have your own gene sets with specific functions, but 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
```

6 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 = 180,
                             exponent = 1,
                             keggGSCs=c("PW_KEGG"),
                             goGSCs = c("GO_MF"),
                             msigdbGSCs = NULL,
                             reportDir = "HTSanalyzerReport",
                             nwAnalysisGenetic = FALSE,
                             nwAnalysisFdr = 0.001)
```

7 Session Info

```
## R version 3.4.2 (2017-09-28)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Debian GNU/Linux 9 (stretch)
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib/libopenblasp-r0.2.19.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=C
##  [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] limma_3.34.1         igraph_1.1.2         GO.db_3.5.0
##  [4] KEGGREST_1.18.0      org.Hs.eg.db_3.5.0   AnnotationDbi_1.40.0
##  [7] IRanges_2.12.0       S4Vectors_0.16.0     Biobase_2.38.0
## [10] BiocGenerics_0.24.0  HTSanalyzeR2_0.99.11 BiocStyle_2.6.0
##
## loaded via a namespace (and not attached):
##  [1] BioNet_1.38.0         Category_2.44.0       bitops_1.0-6
##  [4] bit64_0.9-7           doParallel_1.0.11     RColorBrewer_1.1-2
##  [7] httr_1.3.1            rprojroot_1.2         tools_3.4.2
## [10] backports_1.1.1       R6_2.2.2              DT_0.2
## [13] affyio_1.48.0         cellHTS2_2.42.0       DBI_0.7
## [16] lazyeval_0.2.1        colorspace_1.3-2      curl_3.0
## [19] bit_1.1-12            compiler_3.4.2        preprocessCore_1.40.0
## [22] graph_1.56.0          colourpicker_1.0      bookdown_0.5
## [25] scales_0.5.0          DEoptimR_1.0-8        mvtnorm_1.0-7
## [28] robustbase_0.92-8     genefilter_1.60.0     affy_1.56.0
## [31] RBGL_1.54.0           stringr_1.3.0         digest_0.6.15
## [34] splots_1.44.0         rmarkdown_1.8         XVector_0.18.0
## [37] rrcov_1.4-3           pkgconfig_2.0.1       htmltools_0.3.6
## [40] htmlwidgets_1.0       rlang_0.1.4           RSQLite_2.0
## [43] BiocInstaller_1.28.0  shiny_1.0.5           hwriter_1.3.2
## [46] RCurl_1.95-4.8        magrittr_1.5          Matrix_1.2-12
## [49] Rcpp_0.12.16          munsell_0.4.3         vsn_3.46.0
## [52] stringi_1.1.7         yaml_2.1.16           MASS_7.3-47
## [55] zlibbioc_1.24.0       plyr_1.8.4            grid_3.4.2
## [58] blob_1.1.0            shinydashboard_0.6.1  miniUI_0.1.1
## [61] lattice_0.20-35       splines_3.4.2         Biostrings_2.46.0
## [64] annotate_1.56.1       locfit_1.5-9.1        knitr_1.17
## [67] codetools_0.2-15      XML_3.98-1.9          evaluate_0.10.1
## [70] data.table_1.10.4-3   prada_1.54.0          png_0.1-7
## [73] httpuv_1.3.5          foreach_1.4.4         gtable_0.2.0
## [76] ggplot2_2.2.1         mime_0.5              xtable_1.8-2
## [79] Rmpfr_0.6-1           survival_2.41-3       pcaPP_1.9-72
## [82] tibble_1.3.4          iterators_1.0.8       RankProd_3.4.0
## [85] memoise_1.1.0         cluster_2.0.6         gmp_0.5-13.1
## [88] GSEABase_1.40.1
```

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.

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.