Gene set enrichment and network analyses of high-throughput screens using *HTSanalyzeR*

Xin Wang[†], Camille Terfve[†], John Rose and Florian Markowetz

December 2, 2010

Contents

1 Introduction

In recent years several technological advances have pushed gene perturbation screens to the forefront of functional genomics. Combining high-throughput screening (HTS) techniques with rich phenotypes enables researchers to observe detailed reactions to experimental perturbations on a genome-wide scale. This makes HTS one of the most promising tools in functional genomics.

Although the phenotypes in HTS data correspond to single genes, it becomes more and more important to analyze them in the context of cellular pathways and networks to understand how genes work together. Network analysis of HTS data depends on the dimensionality of the phenotypic read-out [9]. While specialised analysis approaches exist for high-dimensional phenotyping [5], analysis approaches for low-dimensional screens have so far been spread out over diverse softwares and online tools like DAVID [7] or gene set enrichment analysis (GSEA [14]).

Here we provide an integrated analysis package for HTS data that contains gene set and network analysis approaches commonly used in many papers as reviewed by [9]. *HTSanalyzeR* is written in R [11] and freely available via the Bioconductor project [6]. The software interfaces directly with existing HTS pre-processing packages like cellHTS2 [4] or RNAither [12]. Additionally, our software is in the process of being fully integrated in a web-interface for the analysis of HTS data [10] and will thus be easily accessible to non-programmers.

2 An overview of HTSanalyzeR

HTSanalyzeR takes as input HTS data that has already undergone preprocessing and quality control (e.g. by using cellHTS2). It then functionally annotates the hits by gene set enrichment and network analysis approaches (see Figure 1 for an overview).

Gene set analysis. *HTSanalyzeR* implements two approaches: (i) hypergeometric tests for surprising overlap between hits and gene sets, and (ii) cut-off free gene set enrichment analysis which measures if a gene set shows a concordant trend to stronger phenotypes. *HTSanalyzeR* uses gene sets from

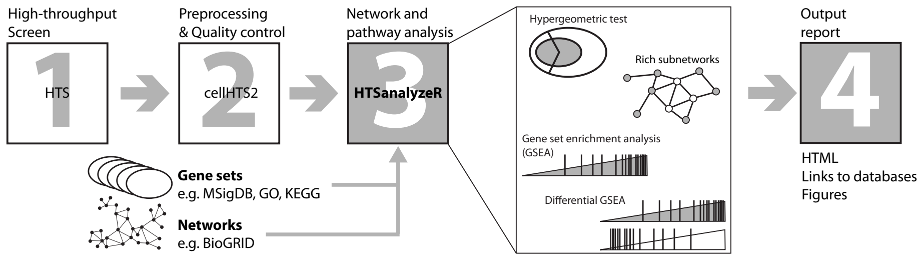

Figure 1: *HTSanalyzeR* takes as input HTS data that has already been preprocessed, normalized and quality checked, e.g. by cellHTS2. *HTSanalyzeR* then combines the HTS data with gene sets and networks from freely available sources and performs three types of analysis: (i) hypergeometric tests for overlap between hits and gene sets, (ii) gene set enrichment analysis (GSEA) for concordant trends of a gene set in one phenotype, (iii) differential GSEA to identify gene sets with opposite trends in two phenotypes, and (iv) identification of subnetworks enriched for hits. The results are provided to the user as figures and HTML tables linked to external databases for annotation.

MSigDB [14], the Gene Ontolology [1] and KEGG [8]. The accompanying vignette explains how user-defined gene sets can easily be included.

Network analysis. In a complementary approach strong hits are mapped to a network and enriched subnetworks are identified. Networks can come from different sources, especially protein interaction networks are often used. In *HTSanalyzeR* we use networks defined in the BioGRID database [13], but other user-defined networks can easily be included in the analysis. To identify rich subnetworks, we use the BioNet package [2], which in its heuristic version is fast and produces close-to-optimal results.

Comparing phenotypes. A goal we expect to become more and more important in the future is to compare phenotypes for the same genes in different cellular conditions. *HTSanalyzeR* supports comparative analyses for gene sets and networks. Differentially enriched gene sets are computed by comparing GSEA enrichment scores or alternatively by a Wilcoxon test statistic. Subnetworks rich for more than one phenotype can easily be found with BioNet [2].

Parameters and report. Each of these analysis methods depends on several input parameters. While every one of them can be changed in the package, *HTSanalyzeR* also implements a standard analysis option using default parameters that we have found to work well in many applications. Results are presented in an HTML format similar to cellHTS2. Overrepresentation and enrichment results are presented as tables, where gene sets are linked to their descriptions at EBI and KEGG pages. GSEA results are accompanied by enrichment plots similar to the ones in [14].

As a demonstration, in this vignette, we introduce how to perform these analyses on an RNAi screen data set in *cellHTS2* format. For other biological data sets, the users can design their own classes, methods and pipelines very easily based on this package.

The packages below need to be loaded before we start the demonstration:

```
> library(HTSanalyzeR)
> library(GSEABase)
> library(cellHTS2)
> library(org.Dm.eg.db)
> library(GO.db)
> library(KEGG.db)
```

3 Preprocessing of high-throughput screens

In this section, we use RNA interference screens as an example to demonstrate how to prepare data for the enrichment and network analyses. The high-throughput screen data set we use here results from a genome-wide RNAi analysis of growth and viability in Drosophila cells [3]. This data set can be found in the package *cellHTS2* ([4]). Before the high-level functional analyses, we need a configured, normalized and annotated cellHTS object that will be used for the networks analysis. This object is then scored to be used in the gene set over-representation part of this analysis. Briefly, the data consists in a series of text files, one for each microtiter plate in the experiment, containing intensity reading for a luciferase reporter of ATP levels in each well of the plate.

The first data processing step is to read the data files and build a cellHTS object from them (performed by the *readPlateList* function).

```
> experimentName <- "KcViab"
> dataPath <- system.file(experimentName, package = "cellHTS2")
> x <- readPlateList("Platelist.txt", name = experimentName,
+ path = dataPath,verbose=TRUE)
```

Then, the object has to be configured, which involves describing the experiment and, more importantly in our case, the plate configuration (i.e. indicating which wells contain samples or controls and which are empty or flagged as invalid).

```
> x <- configure(x, descripFile = "Description.txt", confFile =
+ "Plateconf.txt", logFile = "Screenlog.txt", path = dataPath)
```

Following configuration, the data can be normalized, which is done in this case by substracting from each raw intensity measurement the median of all sample measurements on the same plate.

```
> xn <- normalizePlates(x, scale = "multiplicative", log = FALSE,
+ method = "median", varianceAdjust = "none")
```

In order to use the values in this package, we need to associate each measurement with a meaningful identifier, which can be done by the *annotate* function. In this case, the function will associate with each sample well a flybaseCG identifier, which can be converted subsequently into any identifiers that we might want to use. There are many ways to perform this tasks, for example using our *preprcess* function (in the next section), using a Bioconductor annotation package or taking advantage of the bioMaRt package functionalities. These normalized and annotated values can then be used for the network analysis part of this vignette.

```
> xn <- annotate(xn, geneIDFile = "GeneIDs_Dm_HFA_1.1.txt",
+ path = dataPath)
```

For the gene set over-representation part of this vignette, we choose to work on data that has been scored and summarized. These last processing steps allow us to work with values that have been standardized across samples, resulting in a robust z-score which is indicative of how much the phenotype associated with one condition differs from the bulk. This score effectively quantifies how different a measurement is from the median of all measurements, taking into account the variance (or rather in this case the median absolute deviation) across measurements, therefore reducing the spread of the data. This seems like a sensible measure to be used in gene set over-representation, especially for the GSEA, since it is more readily interpretable and comparable than an absolute phenotype.

```
> xsc <- scoreReplicates(xn, sign = "-", method = "zscore")
```

Moreover the summarization across replicates produces only one value for each construct tested in the screen, which is what we need for the over-representation analysis.

```
> xsc <- summarizeReplicates(xsc, summary = "mean")
```

```
> xsc
```

```
cellHTS (storageMode: lockedEnvironment)
assayData: 21888 features, 1 samples
  element names: score
phenoData
  sampleNames: 1
  varLabels and varMetadata description:
    replicate: Replicate number
    assay: Biological assay
  additional varMetadata: channel
featureData
  featureNames: 1, 2, ..., 21888  (21888 total)
  fvarLabels and fvarMetadata description:
    plate: Plate number
    well: Well ID
    ...: ...
    GeneID: GeneID
    (5 total)
experimentData: use 'experimentData(object)'
state:  configured = TRUE
        normalized = TRUE
        scored = TRUE
        annotated = TRUE
Number of plates: 57
Plate dimension: nrow = 16, ncol = 24
Number of batches: 1
Well annotation: sample other neg pos
  pubMedIds: 14764878
```

For a more detailed description of the preprocessing methods below, please refer to the *cellHTS2* vignette.

4 Gene set overrepresentation and enrichment analysis

4.1 Prepare inputs

To perform gene set enrichment analysis, one must first prepare three inputs:

1. a named numeric vector phenotypes,
2. a character vector of hits, and
3. a list of gene set collections.

First, the phenotypes must be assembled into a vector, annotated, and the replicates must be summarized.

```
> data4enrich <- as.vector(Data(xsc))
> names(data4enrich) <- fData(xsc)[, "GeneID"]
> data4enrich <- data4enrich[which(!is.na(names(data4enrich)))]
```

Then we define the hits as targets displaying phenotypes more than 2 standard deviations away from the mean phenotype, i.e. abs(z-score) > 2.

```
> hits <- names(data4enrich)[which(abs(data4enrich) > 2)]
```

Next, we must define the gene set collections. *HTSanalyzeR* provides facilities which greatly simplify the creation of up-to-date gene set collections. As a simple demonstration, we will test three gene set collections for *Drosophila melanogaster* (see help(annotationConvertor) for details about other species supported): *KEGG* and two *GO* gene set collections. To work properly, these gene set collections must be provided as a named list.

For details on downloading and utilizing gene set collections from Molecular Signatures Database[14], please refer to Appendix B.

```
> GO_MF <- GOGeneSets(species="Dm", ontologies=c("MF"))
> GO_BP <- GOGeneSets(species="Dm", ontologies=c("BP"))
> PW_KEGG <- KeggGeneSets(species="Dm")
> ListGSC <- list(GO_MF=GO_MF, GO_BP=GO_BP, PW_KEGG=PW_KEGG)
```

4.2 Initialize and preprocess

An S4 class *GSCA* (Gene Set Collection Analysis) is developed to do hypergeometric tests for overrepresented gene sets with the list of hits and also performs gene set enrichment analysis (GSEA), as described by Subramanian and colleagues[14].

To begin, an object of class *GSCA* needs to be initialized with a list of gene set collections, phenotypes and hits. If annotations of these input data are not Entrez identifiers, a preprocessing step including input data validation, duplicate removing, annotation conversion and phenotype ordering can be conducted by function *preprocessing*. The user can also build their own preprocessing function according to specific data sets. If input data are 'clean' enough, we can simply choose to perform analyses by function *analyze*.

```
> gsca <- new("GSCA", listOfGeneSetCollections=ListGSC,
+ geneList=data4enrich, hits=hits)
> gsca <- preprocess(gsca, species="Dm", initialIDs="FlybaseCG",
+ keepMultipleMappings=TRUE, duplicateRemoverMethod="max",
+ orderAbsValue=FALSE)
```

4.3 Perform analyses

Then we can use function *analyze* to do the over-representation and enrichment analyses. This function needs an argument called `para`, which is a list of all parameters required to run these analyses. See help("analyze") for the default parameter settings.

```
> gsca<-analyze(gsca, para=list(pValueCutoff=0.05, pAdjustMethod
+ ="BH", nPermutations=100, minGeneSetSize=180, exponent=1))
```

During the enrichment analysis of gene sets, a large number of permutations are required to evaluate the statistical significance. This package supports parallel computing to promote speed based on the *snow* package. To do this, the user simply need to set a cluster called `cluster` before running *analyze*.

```
> library(snow)
> options(htsa.cluster=makeCluster(4, "SOCK"))
```

Please do make sure to stop this cluster and assign 'NULL' to it after the enrichment analysis.

```
> if(is(getOption("cluster"), "cluster")) {
+         stopCluster(getOption("cluster"))
+         options(cluster=NULL)
+ }
```

The output of all analyses stored in slot `result` of the object contains data frames displaying the results for hypergeometric testing and GSEA for each gene set collection, as well as data frames showing the combined results of all gene set collections. Gene sets exhibiting significant p-values for enrichment from both hypergeometric testing and GSEA are displayed in separate tables. Additionally, the output contains dataframes of gene sets exhibiting significant p-values (and significant adjusted p-values) for enrichment from both hypergeometric testing and GSEA.

4.4 Summarize results

A summary method is provided to print summary information about input gene set collections, phenotypes, hits, parameters for hypeogeometric tests and GSEA and results.

```
> summary(gsca)

-No of genes in Gene set collections:
        input above min size
GO_MF    1769             10
GO_BP    2606              7
PW_KEGG   118              1


-No of genes in Gene List:
          input valid duplicate removed converted to entrez
Gene List 13546 13525             12151               11500


-No of hits:
     input preprocessed
Hits  1230         1127


-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 GO_BP PW_KEGG
HyperGeo     5     1       0
GSEA         8     2       0
Both         5     1       0
```

The function *getTopGeneSets* is desinged to retrieve top or all significant gene sets from results of over-representation or GSEA analysis. Basically, the user need to specify the name of results–“HyperGeo.results” or “GSEA.results”, the name(s) of the gene set collection(s) as well as the type of selection–top (by argument `ntop`) or all (by argument `allSig`) significant gene sets.

```
> topGS_GO_MF <- getTopGeneSets(gsca, "GSEA.results", c("GO_MF","PW_KEGG"),
+ allSig=TRUE)

> topGS_GO_MF

$GO_MF
  GO:0003677   GO:0003700   GO:0005515   GO:0008270   GO:0043565 
"GO:0003677" "GO:0003700" "GO:0005515" "GO:0008270" "GO:0043565" 
  GO:0003676   GO:0004252   GO:0005524 
"GO:0003676" "GO:0004252" "GO:0005524" 

$PW_KEGG
named character(0)
```

4.5 Plot significant gene sets

To help the user view GSEA results for a single gene set, the function *viewGSEA* is developed to plot the ranked phenotypes, positions of the gene set and the location of the maximum enrichment score.

```
> viewGSEA(gsca, "GO_MF", topGS_GO_MF[["GO_MF"]][1])
```

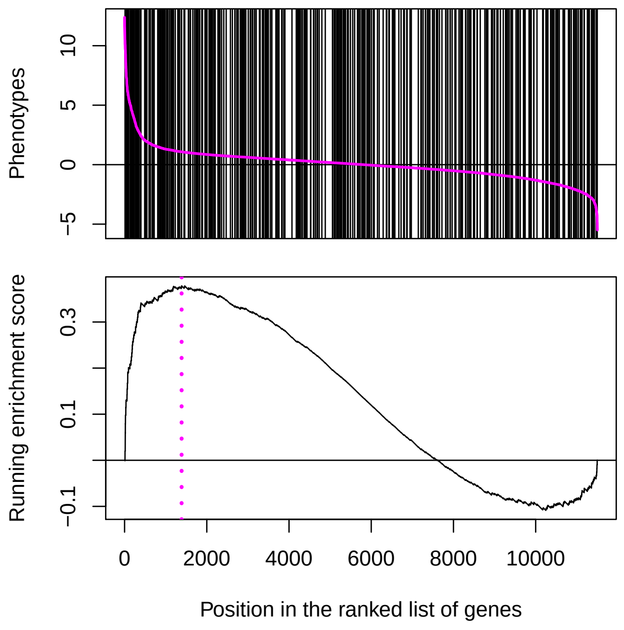

Figure 2: Plot of GSEA result of the most significant gene set of the Molecular Function collection

A plot method (function *plotGSEA*) is also available to plot and store GSEA results of all significant or top gene sets in specified gene set collections in ‘pdf’ or ‘png’ format.

```
> plotGSEA(gsca, gscs=c("GO_BP","GO_MF","PW_KEGG"),
+ ntop=1, filepath=".")
```

4.6 Report results and save objects

The function *report* is used to produce html reports for all gene set analyses.

```
> report(object=gsca, experimentName=experimentName, species="Dm",
+ allSig=TRUE, keggGSCs="PW_KEGG", goGSCs=c("GO_BP", "GO_MF"),
+ reportDir="HTSanalyzerGSCAReport")
```

An index html file containing a summary of all results and hyperlink tables to more detailed results will be generated in the root directory. The other html files will be stored in a subdirectory called “html”. All GSEA plots will be produced in a subdirectory called “image”. All documents or text files such as the files containing significant gene sets of the hypergeometric test results will be stored in a subdirectory called “doc”.

```
> print(dir("HTSanalyzerGSCAReport",recursive=TRUE))
```

```
 [1] "doc/GO:0003677.txt"             "doc/GO:0003700.txt"
 [3] "doc/GO:0005515.txt"             "doc/GO:0006355.txt"
 [5] "doc/GO:0008270.txt"             "doc/GO:0043565.txt"
 [7] "html/enrichment1.html"          "html/enrichment2.html"
 [9] "html/enrichment3.html"          "html/enrichment4.html"
[11] "html/gsea1.html"                "html/gsea2.html"
[13] "html/gsea3.html"                "html/gsea4.html"
[15] "html/hyperg1.html"              "html/hyperg2.html"
[17] "html/hyperg3.html"              "html/hyperg4.html"
[19] "image/blue_cruklogo.gif"        "image/bordercorner.gif"
[21] "image/goatcomputer.png"         "image/gsea_plotsGO:0003676.png"
[23] "image/gsea_plotsGO:0003677.png" "image/gsea_plotsGO:0003700.png"
[25] "image/gsea_plotsGO:0004252.png" "image/gsea_plotsGO:0005515.png"
[27] "image/gsea_plotsGO:0005524.png" "image/gsea_plotsGO:0006355.png"
[29] "image/gsea_plotsGO:0008152.png" "image/gsea_plotsGO:0008270.png"
[31] "image/gsea_plotsGO:0043565.png" "image/gsea_plotsGO:0055085.png"
[33] "image/htsanalyzer.css"          "image/Rlogo.png"
[35] "index.html"
```

To save or load the object of class *GSCA*, we can simply use *save* or *load* similar to other objects of S4 class.

```
> save(gsca, file="./gsca.RData")
> load(file="./gsca.RData")
```

5 Network analysis

As explained above, the data that we use for the network analysis is a configured, normalized and annotated cellHTS object (`xn`). From this object, we extract the normalized data and performs a set of statistical tests for the significance of an observed phenotype by function *cellHTS2OutputStatTests*. We will then aggregate utliple pvalues and map the obtained p-value to an interaction network downloaded from The BioGRID database, and finally use the BioNet package [2] to extract subnetworks enriched with nodes associated with a significant phenotype, from the statistical analysis.

5.1 Prepare input

In the following example, we perform a one sample t-test which tests whether the mean of the observations for each construct is equal to the mean of all sample observations under the null hypothesis. This amounts to testing whether the phenotype associated with a construct is significantly different from the bulk of observations, with the underlying assumption that in a large scale screen (i.e. genome-wide in this case) most constructs are not expected to show a significant effect. We also perform a two-sample t-test, which tests the null hypothesis that two populations have the same mean, where the two populations are a set of observations for each construct and a set of observations for a control population.

To perform those tests, it is mandatory that the samples and controls are labelled in the column `controlStatus` of the fData(xn) dataframe as `sample` and a string specified by the `control` argument of the networkAnalysis function, respectively. Non-parametric tests, such as the *Mann-Whitney U test* and the *Rank Product test*, can also be performed. Both the two samples and the one sample tests are automatically produced, in the case of the t-test and the Mann-Withney U test, by function *cellHTS2OutputStatTests* in the package.

Please be aware that the t-test works under the assumption that the observations are normally distributed and that all of these tests are less reliable when the number of replicates is small. The user should also keep in mind that the one sample t-test assumes that the majority of conditions are not expected to show any significant effect, which is likely to be a dodgy assumption when the size of the screen is small. This test is also to be avoided when the data consists of pre-screened conditions, i.e. constructs that have been selected specifically based on a potential effect.

All three kind of tests can be performed with the 'two sided', 'less' or 'greater' alternative, corresponding to population means (or ranking in the case of the rank product) expected to be different, smaller of larger than the null hypothesis, respectively. For example if your phenotypes consist of cell number and you are looking for constructs that impair cell viability, you might be looking for phenotypes that are smaller than the mean. The `annotationcolumn` argument is used to specify which column of the fData(xn) dataframe contains identifiers for the constructs.

```
> test.stats <- cellHTS2OutputStatTests(cellHTSobject=xn,
+ annotationColumn="GeneID", alternative="two.sided",
+ tests=c("T-test"))
> library(BioNet)
> pvalues <- aggrPvals(test.stats, order=2, plot=FALSE)
```

5.2 Initialize and preprocess

After the p-values of nodes are aggregated, an object of class *NWA* can be created. If phenotypes for these genes are also available, they can be input during the initialization stage. The phenotypes can then be used to highlight nodes in different colors in the identified subnetwork. Another optional argument of the initialize function of class *NWA* is `interactome`, which is the background network of class *graphNEL*. If it is not available, the interactome can be set up later.

```
> data("Biogrid.Dm.interactome")
> nwa <- new("NWA", pvalues=pvalues, interactome=Biogrid.Dm.interactome,
+ phenotypes=data4enrich)
```

The next step is to do preprocessing of input p-values and phenotypes. Similar to class *GSCA*, at the preprocessing stage, the function will also check the validity of input data, remove duplicated genes and convert annotations to Entrez ids. The type of initial identifiers can be specified in the `initialIDs` argument, and will be converted to Entrez gene identifiers which can be mapped to the BioGRID interaction data.

```
> nwa <- new("NWA", pvalues=pvalues, phenotypes=data4enrich)
> nwa <- preprocess(nwa, species="Dm", initialIDs="FlybaseCG",
+ keepMultipleMappings=TRUE, duplicateRemoverMethod="max")
```

To create an interactome for the network analysis, the user can either specify a species to download corresponding network database from *Biogrid*, or input an interaction matrix if the network is already available and in the right format: a matrix with a row for each interaction, and at least the three columns “InteractorA”, “InteractorB” and “InteractionType”, where the interactors are specified by Entrez identifiers..

```
> nwa<-interactome(nwa, species="Dm", reportDir="HTSanalyzerReport",
+ genetic=FALSE)
```

```
> data("Biogrid.Dm.Mat")
> nwa<-interactome(nwa, interactionMatrix=Biogrid.Dm.Mat,
+ genetic=FALSE)
```

```
> nwa@interactome
```

```
A graphNEL graph with undirected edges
Number of Nodes = 6725
Number of Edges = 19689
```

5.3 Perform analysis

After preprocessing of input data and establishment of the interactome, the network analysis can then be performed by calling the same function *analyze* as the object of class *GSCA*. In this function, the argument `fdr` is a parameter for the statistical analysis performed by the BioNet package (see [2] for a guide on how to specify those). The function will produce on your screen a plot showing the fitting of the BioNet model to your distribution of p-values, this is a good plot to check the choice of statistics used in this function.

```
> nwa<-analyze(nwa, fdr=0.001, species="Dm")
```

The *plotSubNet* function produces a figure of the enriched subnetwork, with symbol identifiers as labels of the nodes (if phenotypes have been input during the initialization step).

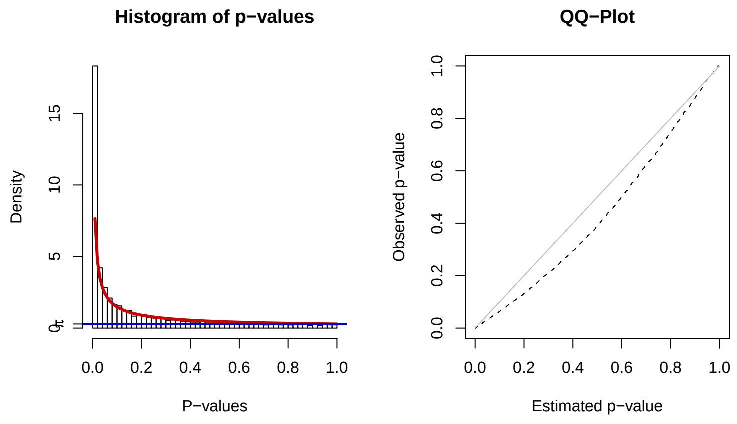

Figure 3: Fitting BUM model to p-values by BioNet.

5.4 Summarize results

Similar to class *GSCA*, a summary method is also available for objects of class *NWA*. The summary includes information about the size of p-value and phenotype vectors before and after preprocessing, the interactome used, parameters and the subnetwork identified by *BioNet*.

```
> summary(nwa)

-p-values:
             input                 valid   duplicate removed
             12170                 12170               12170
converted to entrez      in interactome
             11519                  6094


-Phenotypes:
             input                 valid   duplicate removed
             12170                 12170               12170
converted to entrez      in interactome
             11519                  6094


-Interactome:
```

```
                    name    species genetic node No edge No
Interaction dataset Biogrid <NA>    FALSE   6725    19689


-Parameters for analysis:
            FDR
Parameter 0.001


-Subnetwork identified:
           node No edge No
Subnetwork     119     146
```

5.5 Plot subnetworks

The identified enriched subnetwork will be plot to screen by function *viewSubNet*.

```
> viewSubNet(nwa)
```

To plot and save the subnetwork, we can use function *plotSubNet* with `filepath` and `filename` specified.

```
> plotSubNet(nwa, filepath=".", filename="subnetwork.png")
```

5.6 Report results and save objects

The results of network analysis can be reported by the function *report* in user-specified directory as webpages. An index html file containing a brief summary of the analyses will be generated in the root directory. Another html file including more detailed results will be stored in a subdirectory called "html". One subnetwork figure will be produced in a subdirectory called "image". In addition, a text file containing Entrez ids and gene symbols of the identified subnetwork will be stored in subdirectory "doc".

```
> report(object=nwa, experimentName=experimentName, species="Dm",
+ allSig=TRUE, keggGSCs="PW.KEGG", goGSCs=c("GO.BP", "GO.MF"),
+ reportDir="HTSanalyzerNWReport")
```

To report both results of the enrichment and network analyses, we can use function *reportAll*:

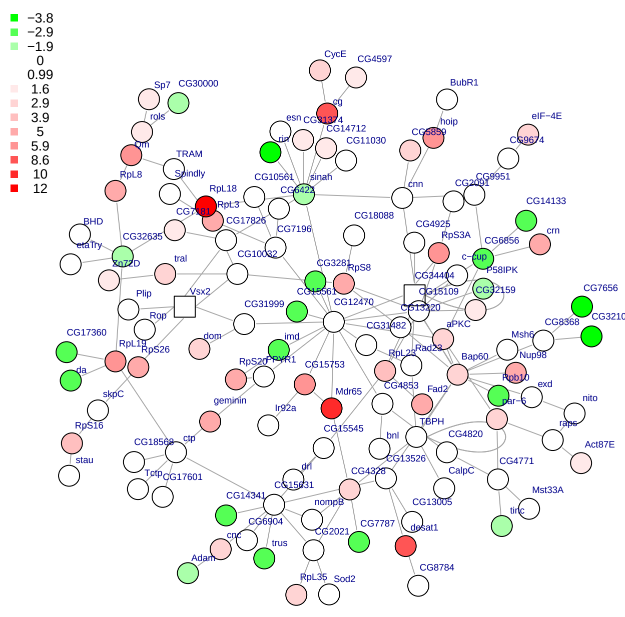

Figure 4: Enriched subnetwork identified by BioNet.

```
> reportAll(gsca=gsca, nwa=nwa, experimentName=experimentName,
+ species="Dm", allSig=TRUE, keggGSCs="PW.KEGG", goGSCs=
+ c("GO.BP", "GO.MF"), reportDir="HTSanalyzerReport")
```

To save and load an object of class *NWA* so that we can reuse all information in the future, we can still use the same way as we did for *GSCA*:

```
> save(nwa, file="./nwa.RData")
> load("./nwa.RData")
```

6 Appendix A: HTSanalyzeR4cellHTS2–A pipeline for cellHTS2 object

All of the above steps can be performed with a unique pipeline function, starting from a normalized, configured and annotated cellHTS object.

First, we need to prepare input data required for analyses just as we introduced in section 3.

```
> data("KcViab.normalized")
> GO_CC<-GOGeneSets(species="Dm",ontologies=c("CC"))
> PW_KEGG<-KeggGeneSets(species="Dm");
> ListGSC<-list(GO_CC=GO_CC,PW_KEGG=PW_KEGG)
```

Then we simply call the function *HTSanalyzeR4cellHTS2*. This will produce a full HTSanalyzeR report, just as if the above steps were performed separately. All the parameters of the enrichment and network analysis steps can be specified as input of this function (see help(HTSanalyzeR4cellHTS2)). Since they are given sensible default values, a minimal set of input parameters is actually required.

```
> HTSanalyzeR4cellHTS2(
+         normCellHTSobject=KcViab.normalized,
+         annotationColumn="GeneID",
+         species=c("Dm"),
+         initialIDs="FlybaseCG",
+         listOfGeneSetCollections=ListGSC,
+         cutoffHitsEnrichment=2,
+         minGeneSetSize=200,
+         keggGSCs=c("PW_KEGG"),
+         goGSCs=c("GO_CC"),
+         reportDir="HTSanalyzerReport"
+ )
```

7 Appendix B: Using MSigDB gene set collections

For experiments in human cell lines, it is often useful to test the gene set collections available at the Molecular Signatures Database (MSigDB; http://www.broadinstitute.org/gsea/msigdb/)[14].

In order to download the gene set collections available through MSigDB, one must first register. After registration, download the desired gmt files into the working directory. Using the *getGmt* and *mapIdentifiers* functions from *GSEABase* importing the gene set collection and mapping the annotations to Entrez IDs is relatively. straightforward

```
> c2<-getGmt(con="c2.all.v2.5.symbols.gmt.txt",geneIdType=
+ SymbolIdentifier(), collectionType=
+ BroadCollection(category="c2"))
```

Once again, for many of the functions in this package to work properly, all gene identifiers must be supplied as Entrez IDs.

```
> c2entrez<-mapIdentifiers(c2, EntrezIdentifier('org.Hs.eg.db'))
```

To create a gene set collection for an object of class *GSCA*, we need to convert from "GeneSetCollection" to a list of gene sets.

```
> collectionOfGeneSets<-geneIds(c2entrez)
> names(collectionOfGeneSets)<-names(c2entrez)
```

8 Appendix C: Performing gene set analysis on multiple phenotypes

When performing hight throughput screens in cell culture-based assays, it is more and more common that multiple phenotypes would be recorded for each condition (such as e.g. number of cells and intensity of a reporter). In these cases, you can perform the enrichment analysis separately on the different lists of phenotypes and try to find gene sets enriched in all of them. In such cases, our package comprises a function called *aggregatePvals* that allows you to aggregate p-values obtained for the same gene set from an enrichment analysis on different phenotypes. This function simply inputs a matrix of p-values with a row for each gene set, and returns aggregated p-values, obtained using either the Fisher or Stouffer methods. The Fisher method combines the p-values into an aggregated chi-squared statistic equal to -2.sum(log(Pk)) were we have k=1,..,K pvalues independently distributed as uniform on the unit interval under the null hypothesis. The resulting p-values is calculated by comparing this chi squared statistic to a chi squared distribution with 2K degrees of freedom. The Stouffer method computes a z statistic assuming that the sum of the quantiles (from a standard normal distribution) corresponding to the p-values are distributed as N(0,K).

However, it is possible that the phenotypes that are measured are expected to show opposite behaviors (e.g. when measuring the number of cells and a reporter for apoptosis). In these cases, we provide two methods to detect gene sets that are associated with opposite patterns of a pair of phenotypic responses. The first method (implemented in the functions *pairwiseGsea* and *pairwiseGseaPlot*) is a modification of the GSEA method by [14]. Briefly, the enrichment scores are computed separately on both phenotype lists, and the absolute value of the difference between the two enrichment scores is compared to permutation-based scores obtained by computing the difference in enrichment score between the two lists when the gene labels are randomly shuffled. This method can only be applied if both phenotypes are measured on the same set of conditions (i.e. the gene labels are the same in both lists, although their associated phenotypes might be very different).

The second method, implemented in the function *pairwisePhenoMannWith*, performs a Mann-Whitney test for shift in location of genes from gene sets, on a pair of phenotypes. The Mann-Whitney test is a non-parametrical equivalent to a two samples t-test (equivalent to a Wilcoxon rank sum test). It looks for gene sets that are whose phenotype distribution is located around two different values in the two phenotypes list, rather than spread on the whole list in both lists. Please be aware that this test should be applied on phenotypes that are on the same scale. If you compare a number of cells (e.g. thousands of cells) to a percentage of cells expressing a marker for example, you will always find a difference in the means of the two populations of phenotypes, whatever the genes in those populations. However, it is very common in high throughput experiments that some sort of internal control is available (e.g. phenotype of the wild type cell line, with no RNAi). A simple way to obtain the different phenotypes on similar scales is therefore to use as phenotypes the raw measurements divided by their internal control counterpart.

Session info

This document was produced using:

```
> toLatex(sessionInfo())
```

- R version 2.11.1 (2010-05-31), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_GB.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB.UTF-8`, `LC_COLLATE=en_GB.UTF-8`, `LC_MONETARY=C`, `LC_MESSAGES=en_GB.UTF-8`, `LC_PAPER=en_GB.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_GB.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, tools, utils
- Other packages: akima 0.5-4, annotate 1.26.1, AnnotationDbi 1.10.2, Biobase 2.8.0, BioNet 1.6.5, cellHTS2 2.12.0, DBI 0.2-5, genefilter 1.30.0, GO.db 2.4.1, graph 1.26.0, GSEABase 1.10.0, HTSanalyzeR 2.1.0, hwriter 1.2, KEGG.db 2.4.1, lattice 0.18-8, locfit 1.5-6, org.Dm.eg.db 2.4.1, RBGL 1.24.0, RColorBrewer 1.0-2, RSQLite 0.9-2, splots 1.14.0, vsn 3.16.0
- Loaded via a namespace (and not attached): affy 1.26.1, affyio 1.16.0, biomaRt 2.4.0, Category 2.14.0, igraph 0.5.4, limma 3.4.4, MASS 7.3-6, prada 1.24.0, preprocessCore 1.10.0, RCurl 1.4-3, rrcov 1.1-00, splines 2.11.1, stats4 2.11.1, survival 2.35-8, XML 3.1-1, xtable 1.5-6

References

[1] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., et al (2000). Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, **25**(1), 25–29. 3

[2] Beisser, D., Klau, G. W., Dandekar, T., Müller, T., Dittrich, M. T. (2010) BioNet: an R-Package for the functional analysis of biological networks. *Bioinformatics*, **26**, 1129-1130 3, 13, 15

[3] Boutros, M., Kiger, A. A., Armknecht, S., Kerr, K., Hild, M., et al (2004). Genome-wide RNAi analysis of growth and viability in drosophila cells. *Science*, **303**(5659), 832–835. 4

[4] Boutros, M., Brás, L. P., and Huber, W. (2006). Analysis of cell-based RNAi screens. *Genome Biol*, **7**(7), R66. 2, 4

[5] Fröhlich, H., Beissbarth, T., Tresch, A., Kostka, D., Jacob, J., Spang, R., and Markowetz, F. (2008). Analyzing gene perturbation screens with nested effects models in R and Bioconductor. *Bioinformatics*, **24**(21), 2549–2550. 2

[6] Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., et al (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol*, **5**(10), R80. 2

[7] Huang, D. W., Sherman, B. T., and Lempicki, R. A. (2009). Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nat Protoc*, **4**(1), 44–57. 2

[8] Kanehisa, M., Goto, S., Hattori, M., Aoki-Kinoshita, K. F., et al. (2006). From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res*, **34**(Database issue), D354–D357. 3

[9] Markowetz, F. (2010). How to understand the cell by breaking it: network analysis of gene perturbation screens. *PLoS Comput Biol*, **6**(2), e1000655. 2

[10] Pelz, O., Gilsdorf, M., and Boutros, M. (2010). web-cellHTS2: a web-application for the analysis of high-throughput screening data. *BMC Bioinformatics*, **11**, 185. 2

[11] R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. 2

[12] Rieber, N., Knapp, B., Eils, R., and Kaderali, L. (2009). RNAither, an automated pipeline for the statistical analysis of high-throughput RNAi screens. *Bioinformatics*, **25**(5), 678–679. 2

[13] Stark, C., Breitkreutz, B.-J., Reguly, T., Boucher, L., Breitkreutz, A., and Tyers, M. (2006). BioGRID: a general repository for interaction datasets. *Nucleic Acids Res*, **34**(Database issue), D535–D539. 3

[14] Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., et al. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A*, **102**(43), 15545–15550. 2, 3, 4, 7, 8, 20, 21