Gene set enrichment and network analysis of high-throughput RNAi screen data using *HTSanalyzeR*

John Rose, Camille Terfve, Xin Wang and Florian Markowetz

November 28, 2010

Contents

1 Introduction

The package *HTSanalyzeR* facilitates integrated high-level analysis of biological data sets such as RNAi high-throughput screening (HTS) data. In particular, this package allows for efficient gene set enrichment and network analyses. Gene set enrichment analysis is performed using both hypergeometric testing and the Gene Set Enrichment Analysis approach reported by Subramanian and colleagues[4]. The network analysis functionalities make use of the BioNet package[1] to identify enriched subnetworks.

Based on the generic classes and methods for GSEA and network analysis, we also designed a pipeline specifically for RNA interference screen data. As we all know, there are various packages (e.g. *cellHTS2*, *RNAither*) for normalization, summarization, and annotation of RNAi HTS raw data, but, to date, there is little support for higher-level analyses of such data. This package adapts a number of methods used for high-level microarray analysis for application to RNA HTS data.

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, users can design their own classes, methods and pipelines very easily based on this package.

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

2 Preprocessing cell-based RNAi screen data with *cellHTS2*

The high-throughput RNAi data set that 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* ([2]). 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). 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). 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. 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 *drosoAnnotationConvertor* function, using a Bioconductor annotation package or taking advantage of the bioMaRt package functionalities. These normalized and annotated values can now be used for the network analysis part of this vignette.

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. 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.

```
> experimentName <- "KcViab"
> dataPath <- system.file(experimentName, package = "cellHTS2")
> x <- readPlateList("Platelist.txt", name = experimentName,
+     path = dataPath, verbose = TRUE)
> x <- configure(x, descripFile = "Description.txt", confFile = "Plateconf.txt",
+     logFile = "Screenlog.txt", path = dataPath)
> xn <- normalizePlates(x, scale = "multiplicative", log = FALSE,
+     method = "median", varianceAdjust = "none")
> xn <- annotate(xn, geneIDFile = "GeneIDs_Dm_HFA_1.1.txt",
+     path = dataPath)
> xsc <- scoreReplicates(xn, sign = "-", method = "zscore")
> xsc <- summarizeReplicates(xsc, summary = "mean")
```

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

3 Gene set overrepresentation and enrichment analysis

3.1 Preparing inputs

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

1. A named vector of numeric phenotypes.
2. A vector of hits
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.

```
> data.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 4 gene set collections for Drosophila melanogaster (see help(annotationConvertor) for details about other species supported): KEGG and all 3 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[4], please refer to Appendix B.

```
> GO.MF <- GOGeneSets(species = "Dm", ontologies = c("MF"))
> GO.BP <- GOGeneSets(species = "Dm", ontologies = c("BP"))
> GO.CC <- GOGeneSets(species = "Dm", ontologies = c("CC"))
> PW.KEGG <- KeggGeneSets(species = "Dm")
> gsc.list <- list(GO.MF = GO.MF, GO.BP = GO.BP, GO.CC = GO.CC,
+     PW.KEGG = PW.KEGG)
```

3.2 Performing analyses

A 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[4].

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*. Users 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 = gsc.list,
+     geneList = data4enrich, hits = data.hits)
> gsca <- preprocess(gsca, species = "Dm", initialIDs = "FlybaseCG",
+     keepMultipleMappings = TRUE, duplicateRemoverMethod = "max",
+     orderAbsValue = FALSE)
```

```
> library(snow)
> options(htsa.cluster = makeCluster(4, "SOCK"))
> gsca <- analyze(gsca, para = list(pValueCutoff = 0.05,
+     pAdjustMethod = "BH", nPermutations = 1000, minGeneSetSize = 50,
+     exponent = 1))
> if (is(getOption("htsa.cluster"), "cluster")) {
+     stopCluster(getOption("htsa.cluster"))
+     options(htsa.cluster = NULL)
+ }
```

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, users simply need to set a cluster called `htsa.cluster` before running *analyze*. But please do make sure to stop this cluster and assign a NULL value to it after the enrichment analysis.

The output 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. Additionally, 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.

A summary method is provided to print summary information about input gene set collections, phenotypes, hits, parameters for hypeogeometric tests and GSEA and results. A plot method is also available to plot all significant or top gene sets in specified gene set collections.

```
> summary(gsca)
> plotGSEA(gsca, gscs = c("GO.BP", "GO.MF", "GO.CC", "PW.KEGG"),
+     allSig = TRUE, filepath = ".")
```

4 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 [1] to extract subnetworks enriched with nodes associated with a significant phenotype, from the statistical analysis.

In the following example, we perform a one sample t-test which tests wether the mean of the observations for each construct is equal to the median 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. Other types of tests can be performed, namely a "Mann-Whitney" test (one or two sample non-parametric equivalent of the t-test), or a rank product test. Both the two samples and the one sample tests are automatically produced, in the case of the t-test and the Mann-Withney test.

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.

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.

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 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.

To create an interactome for the network analysis, users 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..

After preprocessing of input data and establishment of the interactome, the network analysis can be performed by calling the same function *analyze*. In this function, the argument `fdr` is a parameter for the statistical analysis performed by the BioNet package (see [1] 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 look at to check the order parameter and choice of statistics used in this function.

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

```
> library(BioNet)
> test.stats <- cellHTS2OutputStatTests(cellHTSobject = xn,
+     annotationColumn = "GeneID", alternative = "two.sided",
+     tests = c("T-test"))
> pvalues <- aggrPvals(test.stats, order = 2, plot = FALSE)
> nwa <- new("NWA", pvalues = pvalues, phenotypes = data4enrich)
> nwa <- preprocess(nwa, species = "Dm", initialIDs = "FlybaseCG",
+     keepMultipleMappings = TRUE, duplicateRemoverMethod = "max")
> nwa <- interactome(nwa, species = "Dm", reportDir = "HTSanalyzerReport",
+     genetic = FALSE)
> nwa <- analyze(nwa, fdr = 0.001, species = "Dm")
> plotSubNet(nwa, filepath = ".", filename = "subnetwork.png")
```

5 Producing the HTML report

The enrichment analysis and network analysis can be reported seperately (by function *report*) or together (by function *reportAll*) in user-specified directory as webpages. 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 images including GSEA plots and subnetwork figure 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”.

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

6 Appendix A: 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/)[4].

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)
```

7 Appendix B: Using the *HTSanalyzeR4cellHTS2* function

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

First, we need to create the cellHTS object and the gene set collection list, and load the relevant packages.

```
> library(org.Dm.eg.db)
> library(GO.db)
> library(KEGG.db)
> experimentName <- "KcViab"
> dataPath <- system.file(experimentName, package = "cellHTS2")
> x <- readPlateList("Platelist.txt", name = experimentName,
+     path = dataPath)
> x <- configure(x, descripFile = "Description.txt", confFile = "Plateconf.txt",
+     logFile = "Screenlog.txt", path = dataPath)
> xn <- normalizePlates(x, scale = "multiplicative", log = FALSE,
+     method = "median", varianceAdjust = "none")
> xn <- annotate(xn, geneIDFile = "GeneIDs_Dm_HFA_1.1.txt",
+     path = dataPath)
> GO.CC <- GOGeneSets(species = "Dm", ontologies = c("CC"))
> PW.KEGG <- KeggGeneSets(species = "Dm")
> gsc.list <- 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.

```
> cellHTS2DrosoData <- xn
> HTSanalyzeR4cellHTS2(normCellHTSobject = cellHTS2DrosoData,
+     annotationColumn = "GeneID", species = c("Dm"), initialIDs = "FlybaseCG",
+     listOfGeneSetCollections = gsc.list, cutoffHitsEnrichment = 2,
+     minGeneSetSize = 20, keggGSCs = "PW.KEGG", goGSCs = c("GO.CC"),
+     reportDir = "HTSanalyzerReport")
```

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 [4]. 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.

References

[1] D. Beisser, G.W. Klau, T. Dandekar, T. Mueller, and M. Dittrich. Bionet: an r-package for the functional analysis of biological networks. *Bioinformatics*, 2010.

[2] M. Boutros, L. Brás, and W. Huber. Analysis of cell-based rnai screens. *Genome Biology*, 7(7):R66, 2006.

[3] M. Boutros, A.A. Kiger, S. Armknecht, K. Kerr, M. Hild, B. Koch, S.A. Haas, et al. Genome-wide rnai analysis of growth and viability in drosophila cells. *Science*, 303(5659):832, 2004.

[4] A. Subramanian, P. Tamayo, V.K. Mootha, S. Mukherjee, B.L. Ebert, M.A. Gillette, A. Paulovich, S.L. Pomeroy, T.R. Golub, E.S. Lander, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545, 2005.