# EnrichmentBrowser: Seamless navigation through combined results of set-based and network-based enrichment analysis

Ludwig Geistlinger

October 14, 2015

## Contents



## 1 Introduction

The *EnrichmentBrowser* package implements essential functionality for the enrichment analysis of gene expression data. The analysis combines the advantages of set-based and network-based enrichment analysis in order to derive high-confidence gene sets and biological pathways that are differentially regulated in the expression data under investigation. Besides, the package facilitates the visualization and exploration of such sets and pathways.
The following instructions will guide you through an end-to-end expression data analysis workflow including:

1. Preparing the data
2. Preprocess the data
3. Differential expression (DE) analysis
4. Defining gene sets of interest
5. Executing individual enrichment methods
6. Combining the results of different methods
7. Visualize and explore the results

All of these steps are modular, i.e. each step can be executed individually and fine-tuned with several parameters. In case you are interested only in a particular step, you are advised to directly jump to the respective section (Let's say, for example, you are at the point where you have differential expression calculated for each gene. Now you are interested whether certain gene functions are enriched for differential regulation. Section *Set-based enrichment*

*analysis* would then be the one you should go for). The last section *Putting it all together* also demonstrates how to wrap the whole workflow into a single function, making use of suitably chosen defaults.

## 2 Reading expression data from file

Typically, the expression data is not already available in *R* but rather has to be read in from file. This can be done using the function `read.eset`, which reads the expression data (`exprs`) along with the phenotype data (`pData`) and feature data (`fData`) into an *ExpressionSet*.

```
> library(EnrichmentBrowser)
> data.dir <- system.file("extdata", package="EnrichmentBrowser")
> exprs.file <- file.path(data.dir, "exprs.tab")
> pdat.file <- file.path(data.dir, "pData.tab")
> fdat.file <- file.path(data.dir, "fData.tab")
> eset <- read.eset(exprs.file, pdat.file, fdat.file)
```

The man pages provide details on file format and the *ExpressionSet* data structure.

```
> ?read.eset
> ?ExpressionSet
```

# 3 Types of expression data

The two major data types processed by the *EnrichmentBrowser* are microarray (intensity measurements) and RNA-seq (read counts) data.

## 3.1 Microarray data

To demonstrate the functionality of the package for microarray data, we consider expression measurements of patients suffering from acute lymphoblastic leukemia [1]. A frequent chromosomal defect found among these patients is a translocation, in which parts of chromosome 9 and 22 swap places. This results in the oncogenic fusion gene BCR/ABL created by positioning the ABL1 gene on chromosome 9 to a part of the BCR gene on chromosome 22. We load the *ALL* dataset

```
> library(ALL)
> data(ALL)
```

and select B-cell ALL patients with and without the BCR/ABL fusion as it has been described previously [2].

```
> ind.bs <- grep("^B", ALL$BT)
> ind.mut <- which(ALL$mol.biol %in% c("BCR/ABL", "NEG"))
> sset <- intersect(ind.bs, ind.mut)
> all.eset <- ALL[, sset]
```

We can now access the expression values, which are intensity measurements on a log-scale for 12,625 probes (rows) across 79 patients (columns).

```
> dim(all.eset)
```

```
Features  Samples
   12625       79
```

```
> exprs(all.eset)[1:4,1:4]
```

```
             01005    01010    03002    04007
1000_at   7.597323 7.479445 7.567593 7.905312
1001_at   5.046194 4.932537 4.799294 4.844565
1002_f_at 3.900466 4.208155 3.886169 3.416923
1003_s_at 5.903856 6.169024 5.860459 5.687997
```

As we often have more than one probe per gene, we compute gene expression values as the average of the corresponding probe values.

```
> all.eset <- probe.2.gene.eset(all.eset)
> head(featureNames(all.eset))
```

```
[1] "5595" "7075" "1557" "643"  "1843" "4319"
```

(Note, that the mapping from probe to gene is done automatically as long as as you have the corresponding annotation package, here the *hgu95av2.db* package, installed. Otherwise, the mapping can be defined in the `fData` slot.)

```
> head(fData(eset))
```

```
            PROBEID ENTREZID
1000_at     1000_at     5595
1010_at     1010_at     5600
1011_s_at 1011_s_at     7531
1013_at     1013_at     4090
1018_at     1018_at     7480
1019_g_at 1019_g_at     7480
```

## 3.2 RNA-seq data

To demonstrate the functionality of the package for RNA-seq data, we consider transcriptome profiles of four primary human airway smooth muscle cell lines in two conditions: control and treatment with dexamethasone [3].

We load the *airway* dataset

```
> library(airway)
> data(airway)
```

and create the ExpressionSet (for further analysis, we remove genes with very low read counts and measurements that are not mapped to an ENSEMBL gene ID).

```
> expr <- assays(airway)[[1]]
> expr <- expr[grep("^ENSG", rownames(expr)),]
> expr <- expr[rowMeans(expr) > 10,]
> air.eset <- new("ExpressionSet", exprs=expr, annotation="hsa")
> dim(air.eset)
```

```
Features  Samples
   16055        8
```

```
> exprs(air.eset)[1:4,1:4]
```

```
                SRR1039508 SRR1039509 SRR1039512 SRR1039513
ENSG00000000003        679        448        873        408
ENSG00000000419        467        515        621        365
ENSG00000000457        260        211        263        164
ENSG00000000460         60         55         40         35
```

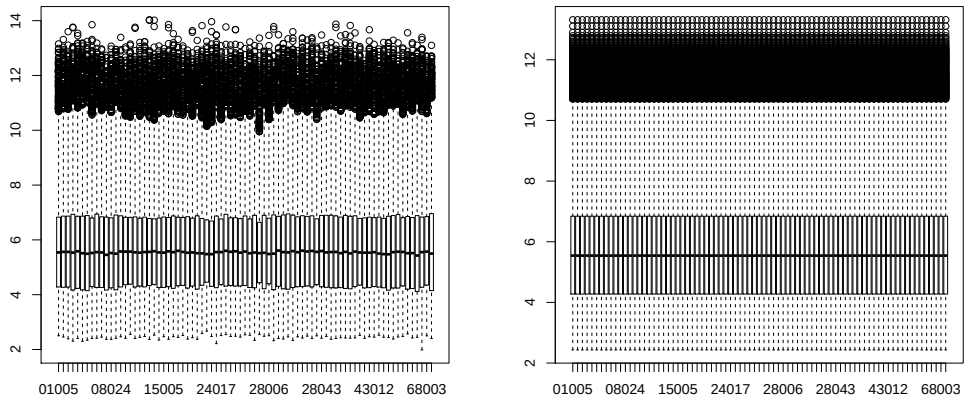
# 4 Normalization

Normalization of high-throughput expression data is essential to make results within and between experiments comparable. Microarray (intensity measurements) and RNA-seq (read counts) data exhibit typically distinct features that need to be normalized for. The function `normalize` wraps commonly used funtionality from *limma* for microarray normalization and from *EDASeq* for RNA-seq normalization. For specific needs that deviate from these standard normalizations, the user should always refer to more specific functions/packages.
Microarray data is expected to be single-channel. For two-color arrays, it is expected here that normalization within arrays has been already carried out, e.g. using `normalizeWithinArrays` from *limma*.

A default quantile normalization based on `normalizeBetweenArrays` from *limma* can be carried out via

```
> before.norm <- exprs(all.eset)
> all.eset <- normalize(all.eset, norm.method="quantile")
> after.norm <- exprs(all.eset)
```

```
> par(mfrow=c(1,2))
> boxplot(before.norm)
> boxplot(after.norm)
```




Note that this is done here for demonstration purpose only, as the ALL data has been already rma-normalized from the authors of the ALL dataset.
RNA-seq data is expected to be raw read counts. Please note that normalization for downstream DE analyis, e.g. with *edgeR* and *DESeq2*, is not ultimately necessary (and in some cases even discouraged) as many of these tools implement specific normalization approaches themselves. See the vignette of *EDASeq*, *edgeR*, and *DESeq2* for details. In case normalization is desired, between-lane normalization to adjust for sequencing depth, can be carried out as demonstrated above for microarray data.

```
> norm.air <- normalize(air.eset, norm.method="quantile")
```

Within-lane normalization to adjust for gene specific effects such as gene length and GC content effect requires to retrieve this information first, e.g. from BioMart or specific *Bioconductor*annotation packages. Both modes are implemented in the *EDASeq* function `getGeneLengthAndGCContent`.

```
> ids <- head(featureNames(air.eset))
> lgc <- EDASeq::getGeneLengthAndGCContent(ids, org="hsa", mode="biomart")
> lgc
```

Using precomputed information for all genes, normalization within and between lanes can then be carried out via

```
> lgc.file <- file.path(data.dir, "air_lgc.tab")
> fData(air.eset) <- read.delim(lgc.file)
> norm.air <- normalize(air.eset, within=TRUE)
```

# 5 Differential expression

Differential expression analysis between sample groups can be performed using the function `de.ana`. As a prerequisite, the phenotype data should contain for each patient a binary group assignment. For the ALL dataset this indicates whether the BCR-ABL gene fusion is present (1) or not (0).

```
> pData(all.eset)$GROUP <- ifelse(all.eset$mol.biol == "BCR/ABL", 1, 0)
> table(pData(all.eset)$GROUP)
```

```
 0  1
42 37
```

For the airway dataset this indicates whether the cell lines have been treated with dexamethasone (1) or not (0).

```
> pData(air.eset)$GROUP <- ifelse(colData(airway)$dex == "trt", 1, 0)
> table(pData(air.eset)$GROUP)
```

```
0 1
4 4
```

Paired samples, or in general sample batches/blocks, can be defined via a `BLOCK` column in the `pData` slot. For the airway dataset the sample blocks correspond to the four different cell lines.

```
> pData(air.eset)$BLOCK <- colData(airway)$cell
> table(pData(air.eset)$BLOCK)
```

```
N052611 N061011 N080611  N61311
      2       2       2       2
```

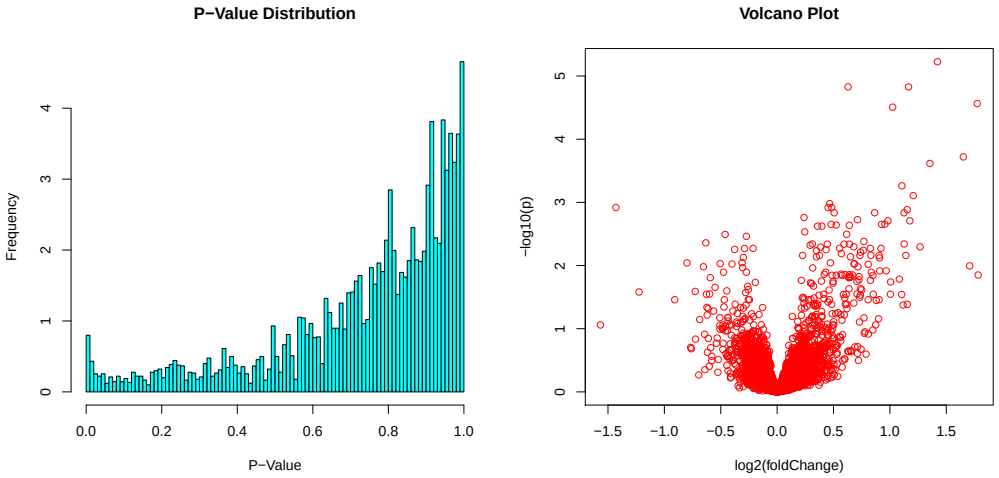
For microarray expression data, the `de.ana` function carries out a differential expression analysis between the two groups based on functionality from the *limma* package. Resulting fold changes and $t$-test derived $p$-values for each gene are appended to the `fData` slot.

```
> all.eset <- de.ana(all.eset)
> head(fData(all.eset), n=4)
```

```
             FC  ADJ.PVAL limma.STAT
5595  0.03880062 0.8615937  0.6590290
7075  0.01732380 0.9588037  0.2457537
1557 -0.05079059 0.6828778 -1.2805512
643  -0.03063407 0.8605821 -0.6646003
```

Raw $p$-values are already corrected for multiple testing (`ADJ.PVAL`) using the method from Benjamini and Hochberg implemented in the function `p.adjust` from the *stats* package.
To get a first overview, we inspect the $p$-value distribution and the volcano plot (fold change against $p$-value).

```
> par(mfrow=c(1,2))
> pdistr(fData(all.eset)$ADJ.PVAL)
> volcano(fData(all.eset)$FC, fData(all.eset)$ADJ.PVAL)
```




The expression change of highest statistical significance is observed for the ENTREZ gene 7525.

```
> fData(all.eset)[ which.min(fData(all.eset)$ADJ.PVAL), ]
```

```
           FC     ADJ.PVAL limma.STAT
7525 1.421774 5.944069e-06   7.019131
```

This turns out to be the YES proto-oncogene 1 (hsa:7525@KEGG).

For RNA-seq data, the `de.ana` function carries out a differential expression analysis between the two groups either based on functionality from *limma* (that includes the `voom` transformation), or alternatively, from the popular *edgeR* or *DESeq2* package. We use here the analysis based on *edgeR* for demonstration.

```
> air.eset <- de.ana(air.eset, de.method="edgeR")
> head(fData(air.eset), n=4)
```

```
                 length        gc           FC     ADJ.PVAL edgeR.STAT
ENSG00000000003   8000 0.4095000 -0.38981443 0.0002054395 17.4402380
ENSG00000000419  23656 0.3982076  0.19817371 0.1083585316  4.0864371
ENSG00000000457  40886 0.4025339  0.02971155 0.8808403672  0.0638300
ENSG00000000460 190985 0.3923816 -0.11753938 0.7193511342  0.3135965
```

Now, we subject the ALL and the airway gene expression data to the enrichment analysis.

# 6 Set-based enrichment analysis

In the following, we introduce how the *EnrichmentBrowser* package can be used to perform state-of-the-art enrichment analysis of gene sets. We consider the ALL and the airway gene expression data as processed in the previous sections. We are now interested whether there are not only single genes that are differentially expressed, but also sets of genes known to work together, e.g. as defined in the Gene Ontology or the KEGG pathway annotation.
The function `get.kegg.genesets`, which is based on functionality from the *KEGGREST* package, downloads all KEGG pathways for a chosen organism as gene sets.

```
> kegg.gs <- get.kegg.genesets("hsa")
```

Analogously, the function `get.go.genesets` defines GO terms of a selected ontology as gene sets.

```
> go.gs <- get.go.genesets(org="hsa", onto="BP", mode="GO.db")
```

User-defined gene sets can be parsed from the GMT file format

```
> gmt.file <- file.path(data.dir, "hsa_kegg_gs.gmt")
> hsa.gs <- parse.genesets.from.GMT(gmt.file)
> length(hsa.gs)
```

```
[1] 39
```

```
> hsa.gs[1:2]
```

```
$hsa05416_Viral_myocarditis
 [1] "100509457" "101060835" "1525"      "1604"      "1605"      "1756"      "1981"
 [8] "1982"      "25"        "2534"      "27"        "3105"      "3106"      "3107"
[15] "3108"      "3109"      "3111"      "3112"      "3113"      "3115"      "3117"
[22] "3118"      "3119"      "3122"      "3123"      "3125"      "3126"      "3127"
[29] "3133"      "3134"      "3135"      "3383"      "3683"      "3689"      "3908"
[36] "4624"      "4625"      "54205"     "5551"      "5879"      "5880"      "5881"
[43] "595"       "60"        "637"       "6442"      "6443"      "6444"      "6445"
[50] "71"        "836"       "841"       "842"       "857"       "8672"      "940"
[57] "941"       "942"       "958"       "959"

$`hsa04622_RIG-I-like_receptor_signaling_pathway`
 [1] "10010"  "1147"   "1432"   "1540"   "1654"   "23586"  "26007"  "29110"  "338376"
[10] "340061" "3439"   "3440"   "3441"   "3442"   "3443"   "3444"   "3445"   "3446"
[19] "3447"   "3448"   "3449"   "3451"   "3452"   "3456"   "3467"   "3551"   "3576"
[28] "3592"   "3593"   "3627"   "3661"   "3665"   "4214"   "4790"   "4792"   "4793"
[37] "5300"   "54941"  "55593"  "5599"   "5600"   "5601"   "5602"   "5603"   "56832"
[46] "57506"  "5970"   "6300"   "64135"  "64343"  "6885"   "7124"   "7186"   "7187"
[55] "7189"   "7706"   "79132"  "79671"  "80143"  "841"    "843"    "8517"   "8717"
[64] "8737"   "8772"   "9140"   "9474"   "9636"   "9641"   "9755"
```

Currently, the following set-based enrichment analysis methods are supported

```
> sbea.methods()
```

```
[1] "ora"   "safe"  "gsea"  "samgs"
```

- ORA: Overrepresentation Analysis (simple and frequently used test based on the hypergeometric distribution [4] for a critical review)
- SAFE: Significance Analysis of Function and Expression (generalization of ORA, includes other test statistics, e.g. Wilcoxon's rank sum, and allows to estimate the significance of gene sets by sample permutation; implemented in the *safe* package)
- GSEA: Gene Set Enrichment Analysis (frequently used and widely accepted, uses a Kolmogorov–Smirnov statistic to test whether the ranks of the $p$-values of genes in a gene set resemble a uniform distribution [5])
- SAMGS: Significance Analysis of Microarrays on Gene Sets (extending the SAM method for single genes to gene set analysis [6])

For demonstration we perform here a basic ORA choosing a significance level $\alpha$ of 0.05.


Figure 1: ORA result view. For each significant gene set in the ranking, the user can select to view (1) a gene report, that lists all genes of a set along with fold change and $t$-test derived p-value, (2) interactive overview plots, such as heatmap, $p$-value distribution, and volcano plot, (3) the pathway in KEGG with differentially expressed genes highlighted in red.

```
> sbea.res <- sbea(method="ora", eset=all.eset, gs=hsa.gs, perm=0, alpha=0.05)
> gs.ranking(sbea.res)
```

```
DataFrame with 5 rows and 4 columns
                                      GENE.SET  NR.GENES NR.SIG.GENES   P.VALUE
                                   <character> <numeric>    <numeric> <numeric>
1 hsa04622_RIG-I-like_receptor_signaling_pathway        54            5   0.00888
2 hsa05130_Pathogenic_Escherichia_coli_infection        43            4   0.01400
3                     hsa04520_Adherens_junction        68            5   0.02610
4                   hsa05206_MicroRNAs_in_cancer       133            8   0.03310
5                     hsa05416_Viral_myocarditis        55            4   0.03720
```

The result of every enrichment analysis is a ranking of gene sets by the corresponding $p$-value. The `gs.ranking` function displays only those gene sets satisfying the chosen significance level $\alpha$.
While such a ranked list is the standard output of existing enrichment tools, the functionality of the *EnrichmentBrowser* package allows visualization and interactive exploration of resulting gene sets far beyond that point. Using the `ea.browse` function creates a HTML summary from which each gene set can be inspected in more detail (this builds on functionality from the *ReportingTools* package). The various options are described in Figure 1.

```
> ea.browse(sbea.res)
```

The goal of the *EnrichmentBrowser* package is to provide the most frequently used enrichment methods. However, it is also possible to exploit its visualization capabilities while using one's own set-based enrichment method. This requires to implement a function that takes the characteristical arguments `eset` (expression data), `gs` (gene sets), `alpha` (significance level), and `perm` (number of permutations). In addition, it must return a numeric vector `ps` storing the resulting $p$-value for each gene set in `gs`. The $p$-value vector must be also named accordingly (i.e. `names(ps) == names(gs)`).
Let us consider the following dummy enrichment method, which randomly renders five gene sets significant and all others insignificant.

```
> dummy.sbea <- function(eset, gs, alpha, perm)
+ {
+         sig.ps <- sample(seq(0,0.05, length=1000),5)
```

```
+         insig.ps <- sample(seq(0.1,1, length=1000), length(gs)-5)
+         ps <- sample(c(sig.ps, insig.ps), length(gs))
+         names(ps) <- names(gs)
+         return(ps)
+ }
```

We can plug this method into sbea as before.

```
> sbea.res2 <- sbea(method=dummy.sbea, eset=all.eset, gs=hsa.gs)
> gs.ranking(sbea.res2)
```

```
DataFrame with 5 rows and 2 columns
                         GENE.SET   P.VALUE
                      <character> <numeric>
1  hsa04068_FoxO_signaling_pathway   0.00676
2    hsa03410_Base_excision_repair   0.01230
3 hsa04066_HIF-1_signaling_pathway   0.01800
4    hsa05217_Basal_cell_carcinoma   0.04020
5     hsa05010_Alzheimer's_disease   0.04510
```

# 7 Network-based enrichment analysis

Having found sets of genes that are differentially regulated in the ALL data, we are now interested whether these findings can be supported by known regulatory interactions. For example, we want to know whether transcription factors and their target genes are expressed in accordance to the connecting regulations. Such information is usually given in a gene regulatory network derived from specific experiments, e.g. using the *GeneNetworkBuilder*, or compiled from the literature ([7] for an example). There are well-studied processes and organisms for which comprehensive and well-annotated regulatory networks are available, e.g. the `RegulonDB` for *E. coli* and `Yeastract` for *S. cerevisiae*. However, in many cases such a network is missing. A first simple workaround is to compile a network from regulations in the KEGG database.
We can download all KEGG pathways of a specified organism via the `download.kegg.pathways` function that exploits functionality from the *KEGGREST* package.

```
> pwys <- download.kegg.pathways("hsa")
```

In this case, we have already downloaded all human KEGG pathways. We parse them making use of the *KEGGgraph* package and compile the resulting gene regulatory network.

```
> pwys <- file.path(data.dir, "hsa_kegg_pwys.zip")
> hsa.grn <- compile.grn.from.kegg(pwys)
> head(hsa.grn)
```

```
     FROM   TO     TYPE
[1,] "3569" "3570" "+"
[2,] "3458" "3459" "+"
[3,] "3458" "3460" "+"
[4,] "1950" "1956" "+"
[5,] "1950" "2064" "+"
[6,] "1950" "3480" "+"
```

Now we are able to perform enrichment analysis based on the compiled network. Currently the following network-based enrichment analysis methods are supported

```
> nbea.methods()
```

```
[1] "ggea"    "nea"     "spia"    "pathnet"
```

- GGEA: Gene Graph Enrichment Analysis (evaluates consistency of known regulatory interactions with the observed expression data [8])
- SPIA: Signaling Pathway Impact Analysis (implemented in the *SPIA* package)
- NEA: Network Enrichment Analysis (implemented in the *neaGUI* package)
- PathNet: Pathway Analysis using Network Information (implemented in the *PathNet* package)

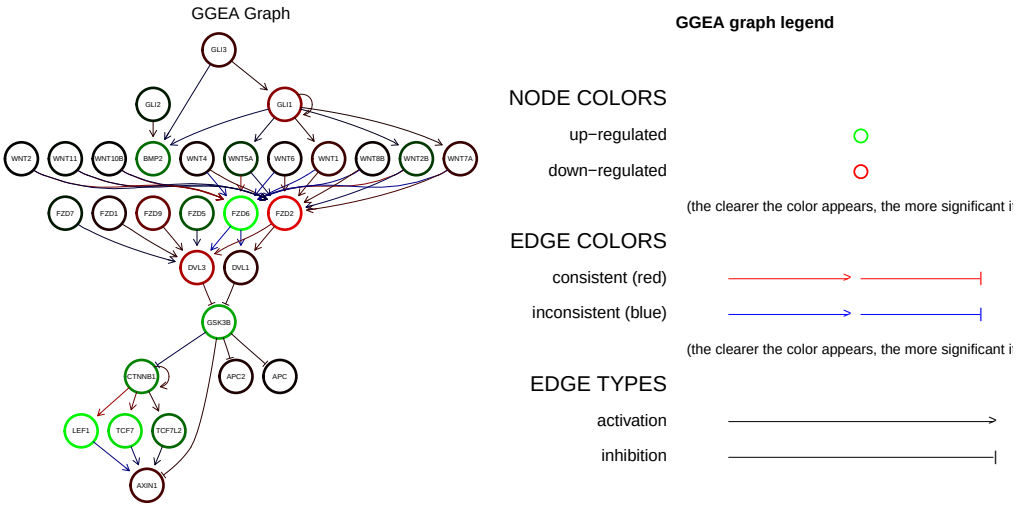
For demonstration we perform here GGEA using the gene regulatory network compiled above.

```
> nbea.res <- nbea(method="ggea", eset=all.eset, gs=hsa.gs, grn=hsa.grn)
> gs.ranking(nbea.res)
```

```
DataFrame with 3 rows and 5 columns
                         GENE.SET   NR.RELS RAW.SCORE NORM.SCORE   P.VALUE
                       <character> <numeric> <numeric>  <numeric> <numeric>
1       hsa05416_Viral_myocarditis         9      3.88      0.431     0.011
2 hsa04390_Hippo_signaling_pathway        61     21.70      0.356     0.015
3               hsa04210_Apoptosis        20      7.69      0.385     0.016
```

The resulting ranking lists for each statistically significant gene set the number of relations (NR.RELS) of the given gene regulatory network that involve a gene set member, the sum of consistencies over all relations (RAW.SCORE), the score normalized by induced network size (NORM.SCORE = RAW.SCORE / NR.RELS), and the statistical significance of each gene set based on a permutation approach.
A GGEA graph for a gene set of interest depicts the consistency of each interaction in the network that involves a gene set member. Nodes (genes) are colored according to expression (up-/down-regulated) and edges (interactions) are colored according to consistency, i.e. how well the interaction type (activation/inhibition) is reflected in the correlation of the observed expression of both interaction partners.

```
> par(mfrow=c(1,2))
> ggea.graph(
+     gs=hsa.gs[["hsa05217_Basal_cell_carcinoma"]],
+     grn=hsa.grn, eset=all.eset)
> ggea.graph.legend()
```




As described in the previous section it is also possible to plug in one's own network-based enrichment method.

# 8 Combining results

Different enrichment analysis methods usually result in different gene set rankings for the same dataset. To compare results and detect gene sets that are supported by different methods, the *EnrichmentBrowser* package allows to combine results from the different set-based and network-based enrichment analysis methods. The combination of results yields a new ranking of the gene sets under investigation either by the average rank across methods or a combined $p$-value using Fisher's method or Stouffer's method [9].
We consider the ORA result and the GGEA result from the previous sections and use the function `comb.ea.results`.

```
> res.list <- list(sbea.res, nbea.res)
> comb.res <- comb.ea.results(res.list)
```

The combined result can be detailedly inspected as before and interactively ranked as depicted in Figure 2.

```
> ea.browse(comb.res, graph.view=hsa.grn, nr.show=5)
```

**COMB - Table of Results**

10 records per page  Search all columns:

| GENE.SET | TITLE | NR.GENES | ORA.RANK | GGEA.RANK | AVG.RANK | ORA.PVAL | GGEA.PVAL | SET.VIEW | PATH.VIEW | GRAPH.VIEW |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | From to | From to | From to | From to | From to | | | |
| hsa05416 | Viral myocarditis | 55 | 5 | 2 | 3 | 0.0372 | 0.006 | | | |
| hsa05130 | Pathogenic Escherichia coli infection | 43 | 2 | 14 | 8 | 0.0140 | 0.243 | | | |
| hsa04514 | Cell adhesion molecules (CAMs) | 107 | 18 | 4 | 11 | 0.3050 | 0.010 | | | |
| hsa04520 | Adherens junction | 68 | 3 | 19 | 11 | 0.0261 | 0.498 | | | |
| hsa05144 | Malaria | 45 | 15 | 7 | 11 | 0.1990 | 0.067 | | | |

Showing 1 to 5 of 5 entries  ← Previous 1 Next →

Figure 2: Combined result view. By clicking on one of the columns (ORA.RANK, ..., GGEA.PVAL) the result can be interactively ranked according to the selected criterion.

# 9 Putting it all together

There are cases where it is necessary to perform some steps of the demonstrated enrichment analysis pipeline individually. However, often it is more convenient to run the complete standardized pipeline. This can be done using the all-in-one wrapper function `ebrowser`. Thus, in order to produce the result page displayed in Figure 2 from scratch, without going through the individual steps listed above, the following call would do the job.

```
> ebrowser(   meth=c("ora", "ggea"),
+         exprs=exprs.file, pdat=pdat.file, fdat=fdat.file,
+         org="hsa", gs=hsa.gs, grn=hsa.grn, comb=TRUE, nr.show=5)
```

# 10 Advanced: Configuring the EnrichmentBrowser

# References

[1] Chiaretti S, Li X, Gentleman R, Vitale A, Vignetti M, and et al. Gene expression profile of adult t-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival. *Blood*, 103(7):2771–8, 2004.

[2] Gentleman R, Carey V, Huber W, Irizarry R, and Dudoit S. Bioinformatics and computational biology solutions using r and bioconductor. *Springer*, New York, 2005.

[3] Himes BE, Jiang X, Wagner P, Hu R, Wang Q, and et al. Rna-seq transcriptome profiling identifies crispld2 as a glucocorticoid responsive gene that modulates cytokine function in airway smooth muscle cells. *PLoS One*, 9(6):e99625, 2014.

[4] Goeman JJ and Buehlmann P. Analyzing gene expression data in terms of gene sets: methodological issues. *Bioinformatics*, 23(8):980–7, 2007.

[5] Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, and et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci USA*, 102(43):15545–50, 2005.

[6] Dinu I, Potter JD, Mueller T, Liu Q, Adewale AJ, and et al. Improving gene set analysis of microarray data by sam-gs. *BMC Bioinformatics*, 8:242, 2007.

[7] Geistlinger L, Csaba G, Dirmeier S, Kueffner R, and Zimmer R. A comprehensive gene regulatory network for the diauxic shift in saccharomyces cerevisiae. *Nucleic Acids Res*, 41(18):8452–63, 2013.

[8] Geistlinger L, Csaba G, Kueffner R, Mulder N, and Zimmer R. From sets to graphs: towards a realistic enrichment analysis of transcriptomic systems. *Bioinformatics*, 27(13):i366–73, 2011.

[9] Kim SC, Lee SJ, Lee WJ, Yum YN, and Kim JH et al. Stouffer's test in a large scale simultaneous hypothesis testing. *PLoS One*, 8(5):e63290, 2013.