Seamless navigation through combined results of set- & network-based enrichment analysis

Ludwig Geistlinger[1]

[1]Center for Computational Biomedicine, Harvard Medical School

July 30, 2023

Abstract

The *EnrichmentBrowser* package implements an analysis pipeline for high-throughput gene expression data as measured with microarrays and RNA-seq. In a workflow-like manner, the package brings together a selection of established Bioconductor packages for gene expression data analysis. It integrates a wide range of gene set and network enrichment analysis methods and facilitates combination and exploration of results across methods.

Package

EnrichmentBrowser 2.30.2

Report issues on https://github.com/lgeistlinger/EnrichmentBrowser/issues

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 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. Preprocessing of 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 in a particular step, you can directly move on to the respective section. For example, if you have differential expression already calculated for each gene, and your are now interested whether certain gene functions are enriched for differential expression, section *Set-based enrichment analysis* would 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 `readSE`, which reads the expression data (`exprs`) along with the phenotype data (`colData`) and feature data (`rowData`) into a *SummarizedExperiment*.

```
library(EnrichmentBrowser)
data.dir <- system.file("extdata", package = "EnrichmentBrowser")
exprs.file <- file.path(data.dir, "exprs.tab")
cdat.file <- file.path(data.dir, "colData.tab")
rdat.file <- file.path(data.dir, "rowData.tab")
se <- readSE(exprs.file, cdat.file, rdat.file)
```

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

```
?readSE
?SummarizedExperiment
```

Note: Previous versions of the *EnrichmentBrowser* used the *ExpressionSet* data structure. The migration to *SummarizedExperiment* in the current release of the *EnrichmentBrowser* is done to reflect recent developments in *Bioconductor*, which discourage use of *ExpressionSet* in favor of *SummarizedExperiment*. Major reasons are the compatibility of *SummarizedExperiment* with operations on genomic regions as well as efficient dealing with big data.

To enable a smooth transition, all functions of the *EnrichmentBrowser* are still accepting also an *ExpressionSet* as input, but are consistently returning a *SummarizedExperiment* as output.

Furthermore, users can always coerce from *SummarizedExperiment* to *ExpressionSet* via

```
eset <- as(se, "ExpressionSet")
```

and vice versa

```
se <- as(eset, "SummarizedExperiment")
```

3 Types of expression data

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

Although RNA-seq has become the *de facto* standard for transcriptomic profiling, it is important to know that many methods for differential expression and gene set enrichment analysis have been originally developed for microarray data.

However, differences in data distribution assumptions (microarray: quasi-normal, RNA-seq: negative binomial) made adaptations in differential expression analysis and, to some extent, also in gene set enrichment analysis necessary.

Thus, we consider two example datasets – a microarray and a RNA-seq dataset, and discuss similarities and differences of the respective analysis steps.

3.1 Microarray data

To demonstrate the functionality of the package for microarray data, we consider expression measurements of patients with acute lymphoblastic leukemia [**?**]. 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 described previously [**?**].

```
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 summarize gene expression values as the average of the corresponding probe values.

```
allSE <- probe2gene(all.eset)
head(rownames(allSE))
```

```
## [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 manually defined in the `rowData` slot.

```
rowData(se)
```

```
## DataFrame with 1000 rows and 1 column
##          ENTREZID
##       <character>
## 3075         3075
## 572           572
## 4267         4267
## 26             26
## 51384       51384
## ...           ...
## 5295         5295
## 2966         2966
## 9140         9140
## 5558         5558
## 1956         1956
```

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 [**?**].

We load the *airway* dataset

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

For further analysis, we remove genes with very low read counts and measurements that are not mapped to an ENSEMBL gene ID.

```
airSE <- airway[grep("^ENSG", rownames(airway)),]
airSE <- airSE[rowSums(assay(airSE)) > 4,]
dim(airSE)
```

```
## [1] 25133     8
```

```
assay(airSE)[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

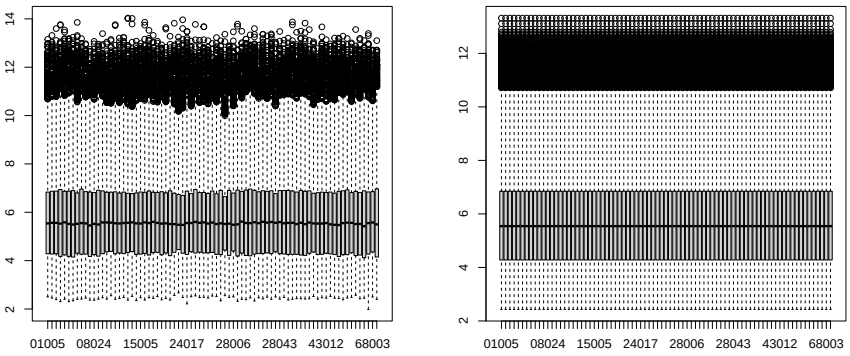
Normalization of high-throughput expression data is essential to make results within and between experiments comparable. Microarray (intensity measurements) and RNA-seq (read counts) data typically show distinct features that need to be normalized for. The function `normalize` wraps commonly used functionality 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 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

```
allSE <- normalize(allSE, norm.method = "quantile")
```

```
par(mfrow=c(1,2))
boxplot(assay(allSE, "raw"))
boxplot(assay(allSE, "norm"))
```



Note that this is only done for demonstration, as the ALL data has been already RMA-normalized by the authors of the ALL dataset.

RNA-seq data is expected to be raw read counts. Note that normalization for downstream DE analysis, 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 for microarray data.

```
airSE <- normalize(airSE, norm.method = "quantile")
```

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

5 Differential expression

The *EnrichmentBrowser* incorporates established functionality from the *limma* package for differential expression analysis between sample groups. This involves the `voom`-transformation when applied to RNA-seq data. Alternatively, differential expression analysis for RNA-seq data can also be carried out based on the negative binomial distribution with *edgeR* and *DESeq2*.

This can be performed using the function `deAna` and assumes some standardized variable names:

- **GROUP** defines the sample groups being contrasted,
- **BLOCK** defines paired samples or sample blocks, as e.g. for batch effects.

For more information on experimental design, see the limma user's guide, chapter 9.

For the ALL dataset, the **GROUP** variable indicates whether the BCR-ABL gene fusion is present (1) or not (0).

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

```
##
##  0  1
## 42 37
```

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

```
airSE$GROUP <- ifelse(airway$dex == "trt", 1, 0)
table(airSE$GROUP)
```

```
##
## 0 1
## 4 4
```

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

```
airSE$BLOCK <- airway$cell
table(airSE$BLOCK)
```

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

For microarray expression data, the `deAna` 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 `rowData` slot.

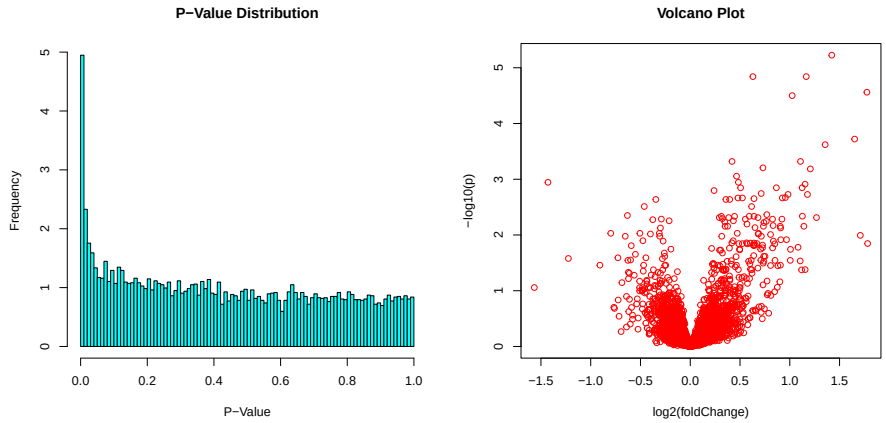
```
allSE <- deAna(allSE, padj.method = "BH")
rowData(allSE)
```

```
## DataFrame with 9054 rows and 4 columns
##              FC limma.STAT      PVAL  ADJ.PVAL
##       <numeric>  <numeric> <numeric> <numeric>
## 5595  0.0391102   0.663788 0.5087392  0.859540
## 7075  0.0165359   0.234715 0.8150313  0.957808
## 1557 -0.0502313  -1.267260 0.2087485  0.687276
## 643  -0.0305410  -0.662652 0.5094628  0.859540
## 1843 -0.4139776  -1.764693 0.0814421  0.512067
## ...         ...        ...       ...       ...
## 6300 -0.0450580  -0.946760  0.346619  0.781975
## 7297 -0.1340087  -1.231891  0.221608  0.700189
## 2246  0.0309975   0.799858  0.426168  0.819762
## 7850 -0.0214471  -0.245232  0.806907  0.957327
## 1593 -0.0126749  -0.254181  0.800009  0.956083
```

Nominal p-values (`PVAL`) are 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(rowData(allSE)$PVAL)
volcano(rowData(allSE)$FC, rowData(allSE)$ADJ.PVAL)
```



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

```
ind.min <- which.min(rowData(allSE)$ADJ.PVAL)
rowData(allSE)[ind.min,]
```

```
## DataFrame with 1 row and 4 columns
##             FC limma.STAT       PVAL    ADJ.PVAL
##      <numeric>  <numeric>  <numeric>   <numeric>
## 7525   1.42179    7.01979 6.5712e-10 5.94956e-06
```

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

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

Here, we use the analysis based on *edgeR* for demonstration.

```
airSE <- deAna(airSE, de.method = "edgeR")
rowData(airSE)
```

```
## DataFrame with 15926 rows and 14 columns
##                         gene_id     gene_name  entrezid   gene_biotype
##                     <character>   <character> <integer>    <character>
## ENSG00000000003 ENSG00000000003        TSPAN6        NA protein_coding
## ENSG00000000419 ENSG00000000419          DPM1        NA protein_coding
## ENSG00000000457 ENSG00000000457         SCYL3        NA protein_coding
## ENSG00000000460 ENSG00000000460      C1orf112        NA protein_coding
## ENSG00000000971 ENSG00000000971           CFH        NA protein_coding
## ...                        ...           ...       ...            ...
## ENSG00000273373 ENSG00000273373  RP5-1074L1.4        NA      antisense
## ENSG00000273382 ENSG00000273382 RP5-1065J22.8        NA      antisense
## ENSG00000273448 ENSG00000273448  RP11-166O4.6        NA        lincRNA
## ENSG00000273472 ENSG00000273472 RP11-102N12.3        NA        lincRNA
## ENSG00000273486 ENSG00000273486 RP11-731C17.2        NA      antisense
##                 gene_seq_start gene_seq_end    seq_name seq_strand
##                      <integer>    <integer> <character>  <integer>
## ENSG00000000003       99883667     99894988           X         -1
## ENSG00000000419       49551404     49575092          20         -1
## ENSG00000000457      169818772    169863408           1         -1
## ENSG00000000460      169631245    169823221           1          1
## ENSG00000000971      196621008    196716634           1          1
## ...                        ...          ...         ...        ...
## ENSG00000273373      110912776    110915625           1          1
## ENSG00000273382      109630593    109633480           1         -1
## ENSG00000273448       66798034     66799370           7          1
## ENSG00000273472      141677682    141679075           4          1
## ENSG00000273486      136556180    136557863           3         -1
##                 seq_coord_system        symbol         FC edgeR.STAT
##                        <integer>   <character>  <numeric>  <numeric>
## ENSG00000000003               NA        TSPAN6 -0.3901002 31.0558140
## ENSG00000000419               NA          DPM1  0.1978022  6.6454709
## ENSG00000000457               NA         SCYL3  0.0291609  0.0929623
```

```
## ENSG00000000460              NA      C1orf112 -0.1243820  0.3832263
## ENSG00000000971              NA           CFH  0.4172901 28.7686093
## ...                         ...           ...        ...        ...
## ENSG00000273373              NA  RP5-1074L1.4 -0.0438722  0.0397087
## ENSG00000273382              NA RP5-1065J22.8 -0.8597567  7.7869742
## ENSG00000273448              NA  RP11-166O4.6  0.0281667  0.0103270
## ENSG00000273472              NA RP11-102N12.3 -0.4642705  1.9010366
## ENSG00000273486              NA RP11-731C17.2 -0.1109445  0.1536285
##                        PVAL   ADJ.PVAL
##                   <numeric>  <numeric>
## ENSG00000000003 0.000232422 0.00217355
## ENSG00000000419 0.027419893 0.07560513
## ENSG00000000457 0.766666551 0.84808859
## ENSG00000000460 0.549659194 0.67996523
## ENSG00000000971 0.000312276 0.00272063
## ...                     ...        ...
## ENSG00000273373   0.8460260   0.901607
## ENSG00000273382   0.0190219   0.057267
## ENSG00000273448   0.9752405   0.984888
## ENSG00000273472   0.1978818   0.328963
## ENSG00000273486   0.7032766   0.802377
```

6 ID mapping

Using genomic information from different resources often requires mapping between different types of gene identifiers. Although primary analysis steps such as normalization and differential expression analysis can be carried out independent of the gene ID type, downstream exploration functionality of the *EnrichmentBrowser* is consistently based on NCBI Entrez Gene IDs. It is thus, in this regard, beneficial to initially map gene IDs of a different type to NCBI Entrez IDs.

The function `idTypes` lists the available ID types for the mapping depending on the organism under investigation.

```
idTypes("hsa")
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"
##  [5] "ENSEMBLTRANS" "ENTREZID"     "ENZYME"       "EVIDENCE"
##  [9] "EVIDENCEALL"  "GENENAME"     "GENETYPE"     "GO"
## [13] "GOALL"        "IPI"          "MAP"          "OMIM"
## [17] "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
## [21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"
## [25] "UCSCKG"       "UNIPROT"
```

ID mapping for the airway dataset (from ENSEMBL to ENTREZ gene ids) can then be carried out using the function `idMap`.

```
head(rownames(airSE))
```

```
## [1] "ENSG00000000003" "ENSG00000000419" "ENSG00000000457" "ENSG00000000460"
## [5] "ENSG00000000971" "ENSG00000001036"
```

```
airSE <- idMap(airSE, org = "hsa", from = "ENSEMBL", to = "ENTREZID")
head(rownames(airSE))
```

```
## [1] "7105"  "8813"  "57147" "55732" "3075"  "2519"
```

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

7 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 in whether predefined sets of genes that are known to work together, e.g. as defined in the Gene Ontology (GO) or the KEGG pathway annotation, are coordinately differentially expressed.

7.1 Obtaining gene sets

The function `getGenesets` can be used to download gene sets from databases such as GO and KEGG. We can use the function to download all KEGG pathways for a chosen organism (here: *Homo sapiens*) as gene sets.

```
kegg.gs <- getGenesets(org = "hsa", db = "kegg")
```

Analogously, the function `getGenesets` can be used to retrieve GO terms of a selected ontology (here: biological process, BP) as defined in the *GO.db* annotation package.

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

If provided a file, the function parses user-defined gene sets from GMT file format. Here, we use this functionality for reading a list of already downloaded KEGG gene sets for *Homo sapiens* containing NCBI Entrez Gene IDs.

```
gmt.file <- file.path(data.dir, "hsa_kegg_gs.gmt")
hsa.gs <- getGenesets(gmt.file)
length(hsa.gs)
```

```
## [1] 39
```

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

```
## $hsa05416_Viral_myocarditis
##  [1] "100509457" "101060835" "1525"      "1604"      "1605"      "1756"
##  [7] "1981"      "1982"      "25"        "2534"      "27"        "3105"
## [13] "3106"      "3107"      "3108"      "3109"      "3111"      "3112"
## [19] "3113"      "3115"      "3117"      "3118"      "3119"      "3122"
## [25] "3123"      "3125"      "3126"      "3127"      "3133"      "3134"
## [31] "3135"      "3383"      "3683"      "3689"      "3908"      "4624"
## [37] "4625"      "54205"     "5551"      "5879"      "5880"      "5881"
## [43] "595"       "60"        "637"       "6442"      "6443"      "6444"
## [49] "6445"      "71"        "836"       "841"       "842"       "857"
## [55] "8672"      "940"       "941"       "942"       "958"       "959"
##
## $`hsa04622_RIG-I-like_receptor_signaling_pathway`
##  [1] "10010"  "1147"   "1432"   "1540"   "1654"   "23586"  "26007"  "29110"
##  [9] "338376" "340061" "3439"   "3440"   "3441"   "3442"   "3443"   "3444"
## [17] "3445"   "3446"   "3447"   "3448"   "3449"   "3451"   "3452"   "3456"
## [25] "3467"   "3551"   "3576"   "3592"   "3593"   "3627"   "3661"   "3665"
## [33] "4214"   "4790"   "4792"   "4793"   "5300"   "54941"  "55593"  "5599"
## [41] "5600"   "5601"   "5602"   "5603"   "56832"  "57506"  "5970"   "6300"
## [49] "64135"  "64343"  "6885"   "7124"   "7186"   "7187"   "7189"   "7706"
```

```
## [57] "79132"  "79671"  "80143"  "841"    "843"    "8517"   "8717"   "8737"
## [65] "8772"   "9140"   "9474"   "9636"   "9641"   "9755"
```

Note #1: Use `getGenesets` with `db = "msigdb"` to obtain gene set collections for 11 different species from the Molecular Signatures Database (MSigDB). Analogously, `getGenesets` with `db = "enrichr"` allows to obtain gene set libraries from the comprehensive Enrichr collection for 5 different species.

Note #2: The `idMap` function can be used to map gene sets from NCBI Entrez Gene IDs to other common gene ID types such as ENSEMBL gene IDs or HGNC symbols as described in Section **??**.

7.2 Set-based enrichment analysis

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

```
sbeaMethods()
```

```
##  [1] "ora"        "safe"       "gsea"       "gsa"        "padog"
##  [6] "globaltest" "roast"      "camera"     "gsva"       "samgs"
## [11] "ebm"        "mgsa"
```

- ORA: Overrepresentation Analysis (simple and frequently used test based on the hypergeometric distribution, see [**?**] for a critical review),
- SAFE: Significance Analysis of Function and Expression (resampling version of ORA, implements additional 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 [**?**]),
- PADOG: Pathway Analysis with Down-weighting of Overlapping Genes (incorporates gene weights to favor genes appearing in few pathways versus genes that appear in many pathways; implemented in the *PADOG* package),
- ROAST: ROtAtion gene Set Test (uses rotation instead of permutation for assessment of gene set significance; implemented in the *limma* and *edgeR* packages for microarray and RNA-seq data, respectively),
- CAMERA: Correlation Adjusted MEan RAnk gene set test (accounts for inter-gene correlations as implemented in the *limma* and *edgeR* packages for microarray and RNA-seq data, respectively),
- GSA: Gene Set Analysis (differs from GSEA by using the maxmean statistic, i.e. the mean of the positive or negative part of gene scores in the gene set; implemented in the *GSA* package),
- GSVA: Gene Set Variation Analysis (transforms the data from a gene by sample matrix to a gene set by sample matrix, thereby allowing the evaluation of gene set enrichment for each sample; implemented in the *GSVA* package)
- GLOBALTEST: Global testing of groups of genes (general test of groups of genes for association with a response variable; implemented in the *globaltest* package),

- SAMGS: Significance Analysis of Microarrays on Gene Sets (extending the SAM method for single genes to gene set analysis [**?**]),
- EBM: Empirical Brown's Method (combines p-values of genes in a gene set using Brown's method to combine p-values from dependent tests; implemented in *EmpiricalBrownsMethod*),
- MGSA: Model-based Gene Set Analysis (Bayesian modeling approach taking set overlap into account by working on all sets simultaneously, thereby reducing the number of redundant sets; implemented in *mgsa*).

See also Appendix **??** for a comprehensive introduction on underlying statistical concepts.

7.3 Guidelines for input and method selection

We recently performed a comprehensive assessment of the availabe set-based enrichment methods, and identified significant differences in runtime and applicability to RNA-seq data, fraction of enriched gene sets depending on the null hypothesis tested, and detection of relevant processes [**?**]. Based on these results, we make practical recommendations on how methods originally developed for microarray data can efficiently be applied to RNA-seq data, how to interpret results depending on the type of gene set test conducted and which methods are best suited to effectively prioritize gene sets with high phenotype relevance:

- for the exploratory analysis of **simple gene lists**, we recommend ORA given its ease of applicability, fast runtime and evident relevance of resulting gene set rankings, provided that input gene list and reference gene list are chosen carefully and remembering ORA's propensity for type I error rate inflation when genes tend to be co-expressed within sets.
- for the analysis of **pre-ranked gene lists** accompanied by gene scores such as fold changes, alternatives to ORA such as pre-ranked GSEA or pre-ranked CAMERA exist.
- for expression-based enrichment analysis on the **full expression matrix**, we recommend providing normalized log2 intensities for microarray data, and logTPMs (or logRPKMs / logFPKMs) for RNA-seq data. When given raw read counts, we recommend to apply a variance-stabilizing transformation such as `voom` to arrive at library-size normalized logCPMs.
- if the question of interest is to test for association of any gene in the set with the phenotype (**self-contained** null hypothesis), we recommend ROAST or GSVA that both test a **directional** hypothesis (genes in the set tend to be either predominantly up- or down-regulated). Both methods can be applied for simple or extended experimental designs, where ROAST is the more natural choice for the comparison of sample groups and also allows one to test a **mixed** hypothesis (genes in the set tend to be differentially expressed, regardless of the direction). The main strength of GSVA lies in its capabilities for analyzing single samples.
- if the question of interest is to test for excess of differential expression in a gene set relative to genes outside the set (**competitive** null hypothesis), which we believe comes closest to the expectations and intuition of most end users when performing GSEA, we recommend PADOG, which is slower to run but resolves major shortcomings of ORA, and has desirable properties for the analyzed criteria and when compared to other competitive methods. However, PADOG is limited to testing a mixed hypothesis in a comparison of two sample groups, optionally including paired samples or sample batches.

Therefore, we recommend the highly customizable SAFE for testing a directional hypothesis or in situations of more complex experimental designs such as comparisons between multiple groups, continuous phenotypes or the presence of covariates.

See also Reference [**?**] for the results of the benchmarking study and the *GSEABenchmarkeR* package for a general framework for reproducible benchmarking of gene set enrichment methods.

7.3.1 Microarray data

Given normalized log2 intensities for the ALL microarray dataset, a basic ORA can be carried out via

```
sbea.res <- sbea(method = "ora", se = allSE, gs = hsa.gs, perm = 0, alpha = 0.1)
gsRanking(sbea.res)
```

```
## DataFrame with 4 rows and 4 columns
##                 GENE.SET  NR.GENES NR.SIG.GENES      PVAL
##              <character> <numeric>    <numeric> <numeric>
## 1 hsa05130_Pathogenic_..        44            5    0.0298
## 2 hsa05206_MicroRNAs_i..       133           10    0.0371
## 3 hsa04622_RIG-I-like_..        55            5    0.0681
## 4 hsa04670_Leukocyte_t..        94            7    0.0810
```

Note that we set `perm = 0` to invoke the classical hypergeometric test without sample permutation, and that we chose a significance level α of 0.1 for demonstration purposes.

7.3.2 RNA-seq data

When analyzing RNA-seq datasets with expression values given as logTPMs (or logRPKMs / logFPKMs), the available set-based enrichment methods can be applied as for microarray data. However, when given raw read counts as for the airway dataset, we recommend to first apply a variance-stabilizing transformation such as `voom` to arrive at library-size normalized logCPMs.

```
airSE <- normalize(airSE, norm.method = "vst")
```

The mean-variance relationship of the transformed data is similar to what is observed for microarray data, simplifying the application of legacy enrichment methods such as GSEA and PADOG to RNA-seq data, and enable the use of fast and established methods.

```
air.res <- sbea(method = "gsea", se = airSE, gs = hsa.gs)
gsRanking(sbea.res)
```

7.4 Result exploration

The result of every enrichment analysis is a ranking of gene sets by the corresponding p-value. The `gsRanking` function displays by default only those gene sets satisfying the chosen significance level α, but we can use

```
gsRanking(sbea.res, signif.only = FALSE)
```

```
## DataFrame with 39 rows and 4 columns
##                   GENE.SET  NR.GENES NR.SIG.GENES      PVAL
```

```
##                <character> <numeric>    <numeric> <numeric>
## 1   hsa05130_Pathogenic_..        44            5    0.0298
## 2   hsa05206_MicroRNAs_i..       133           10    0.0371
## 3   hsa04622_RIG-I-like_..        55            5    0.0681
## 4   hsa04670_Leukocyte_t..        94            7    0.0810
## 5   hsa05100_Bacterial_i..        64            5    0.1130
## ...                   ...       ...          ...       ...
## 35       hsa05218_Melanoma        58            0         1
## 36  hsa05150_Staphylococ..        46            0         1
## 37  hsa03420_Nucleotide_..        41            0         1
## 38  hsa03030_DNA_replica..        33            0         1
## 39  hsa03410_Base_excisi..        27            0         1
```

to obtain the full ranking.

While such a ranked list is the standard output of existing enrichment tools, the *EnrichmentBrowser* package provides visualization and interactive exploration of resulting gene sets far beyond that point. Using the `eaBrowse` function creates a HTML summary from which each gene set can be inspected in detail (this builds on functionality from the *ReportingTools* package).

The various options are described in Figure **??**.

```
eaBrowse(sbea.res)
```

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 DE 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.

7.5 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 (activation/inhibition). Such information is usually given in a gene regulatory network derived from specific experiments or compiled from the literature ([**?**] 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, there are also cases where such a network is missing. A basic workaround is to compile a network from regulations in pathway databases such as KEGG.

```
hsa.grn <- compileGRN(org="hsa", db="kegg")
head(hsa.grn)
```

```
##      FROM    TO          TYPE
## [1,] "10000" "100132074" "-"
## [2,] "10000" "1026"      "+"
## [3,] "10000" "1026"      "-"
## [4,] "10000" "1027"      "-"
## [5,] "10000" "10488"     "+"
## [6,] "10000" "107"       "+"
```

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

```
nbeaMethods()
```

```
## [1] "ggea"        "spia"        "pathnet"     "degraph"     "ganpa"
## [6] "cepa"        "topologygsa" "netgsa"      "neat"
```

- GGEA: Gene Graph Enrichment Analysis (evaluates consistency of known regulatory interactions with the observed expression data [**?**]),
- SPIA: Signaling Pathway Impact Analysis (combines ORA with the probability that expression changes are propagated across the pathway topology; implemented in the *SPIA* package),
- PathNet: Pathway Analysis using Network Information (applies ORA on combined evidence for the observed signal for gene nodes and the signal implied by connected neighbors in the network; implemented in the *PathNet* package),
- DEGraph: Differential expression testing for gene graphs (multivariate testing of differences in mean incorporating underlying graph structure; implemented in the *DEGraph* package),
- TopologyGSA: Topology-based Gene Set Analysis (uses Gaussian graphical models to incorporate the dependence structure among genes as implied by pathway topology; implemented in the *topologyGSA* package),
- GANPA: Gene Association Network-based Pathway Analysis (incorporates network-derived gene weights in the enrichment analysis; implemented in the *GANPA* package),

- CePa: Centrality-based Pathway enrichment (incorporates network centralities as node weights mapped from differentially expressed genes in pathways; implemented in the *CePa* package),
- NetGSA: Network-based Gene Set Analysis (incorporates external information about interactions among genes as well as novel interactions learned from data; implemented in the *NetGSA* package),

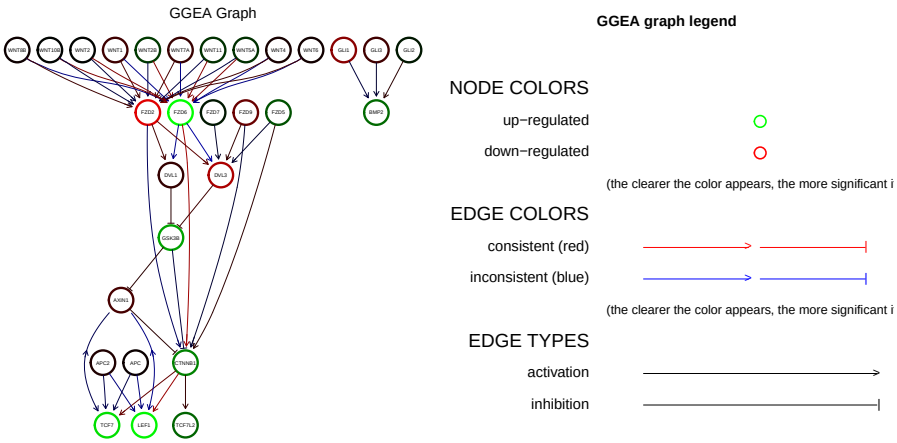
For demonstration, we perform GGEA using the compiled KEGG regulatory network.

```
nbea.res <- nbea(method="ggea", se=allSE, gs=hsa.gs, grn=hsa.grn)
gsRanking(nbea.res)
```

```
## DataFrame with 9 rows and 5 columns
##                 GENE.SET   NR.RELS RAW.SCORE NORM.SCORE      PVAL
##              <character> <numeric> <numeric>  <numeric> <numeric>
## 1 hsa04622_RIG-I-like_..        36     13.70      0.382   0.00200
## 2 hsa05416_Viral_myoca..         7      3.29      0.470   0.00300
## 3 hsa04520_Adherens_ju..        12      4.86      0.405   0.00599
## 4 hsa05217_Basal_cell_..        17      6.60      0.388   0.00899
## 5 hsa04390_Hippo_signa..        66     23.20      0.351   0.00899
## 6 hsa05412_Arrhythmoge..         5      2.14      0.429   0.01900
## 7 hsa05134_Legionellosis        18      6.71      0.373   0.02100
## 8 hsa04621_NOD-like_re..        40     13.90      0.349   0.02600
## 9     hsa04210_Apoptosis        59     20.10      0.341   0.04200
```

The resulting ranking lists, for each statistically significant gene set, the number of relations of the network involving a member of the gene set under study (`NR.RELS`), the sum of consistencies over the relations of the set (`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 depicts the consistency of each interaction in the set. 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))
ggeaGraph(
    gs=hsa.gs[["hsa05217_Basal_cell_carcinoma"]],
    grn=hsa.grn, se=allSE)
ggeaGraphLegend()
```



7.6 User-defined enrichment methods

The goal of the *EnrichmentBrowser* package is to provide frequently used enrichment methods. However, it is also possible to exploit its visualization capabilities with user-defined set-based enrichment methods.

This requires to implement a function that takes the characteristic arguments `se` (expression data) and `gs` (gene sets).

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 also be named accordingly (i.e. `names(ps) == names(gs)`).

Let us consider the following dummy enrichment method, which randomly renders five gene sets significant and the remaining insignificant.

```
dummySBEA <- function(se, gs)
{
        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 = dummySBEA, se = allSE, gs = hsa.gs)
gsRanking(sbea.res2)
```

```
## DataFrame with 5 rows and 2 columns
##                 GENE.SET      PVAL
##              <character> <numeric>
## 1 hsa04622_RIG-I-like_..    0.0111
## 2 hsa05206_MicroRNAs_i..    0.0197
## 3 hsa00790_Folate_bios..    0.0240
## 4 hsa03420_Nucleotide_..    0.0414
## 5 hsa00053_Ascorbate_a..    0.0495
```

As described in the previous section, it is also possible to analogously plug in user-defined network-based enrichment methods into `nbea`.

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 by specified ranking criteria, e.g. the average rank across methods. We consider the ORA result and the GGEA result from the previous sections and use the function `combResults`.

```
res.list <- list(sbea.res, nbea.res)
comb.res <- combResults(res.list)
```

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

```
eaBrowse(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
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 certain steps of the demonstrated enrichment analysis pipeline individually. However, it is often more convenient to run the complete standardized pipeline. This can be done using the all-in-one wrapper function `ebrowser`. For example, the result page displayed in Figure **??** can also be produced from scratch via

```
ebrowser(   meth=c("ora", "ggea"),
        exprs=exprs.file, cdat=cdat.file, rdat=rdat.file,
        org="hsa", gs=hsa.gs, grn=hsa.grn, comb=TRUE, nr.show=5)
```

10 Advanced: configuration parameters

Similar to *R*'s options settings, the *EnrichmentBrowser* uses certain package-wide configuration parameters, which affect the way in which analysis is carried out and how results are displayed. The settings of these parameters can be examined and, to some extent, also changed using the function `configEBrowser`. For instance, the default directory where the *EnrichmentBrowser* writes results to can be updated via

```
configEBrowser(key="OUTDIR.DEFAULT", value="/my/out/dir")
```

and examined via

```
configEBrowser("OUTDIR.DEFAULT")

## [1] "/my/out/dir"
```

Note that changing these defaults should be done with care, as inappropriate settings might impair the package's functionality. The complete list of incorporated configuration parameters along with their default settings can be inspected via

```
?configEBrowser
```

11 For non-R users: command line invocation

The package source contains two scripts in `inst/scripts` to invoke the EnrichmentBrowser from the command line using *Rscript*.

The `de_rseq.R` script is a lightweight wrapper script to carry out differential expression analysis of RNA-seq data either based on *limma* (using the `voom`-transformation), *edgeR*, or *DESeq2*.

The `eBrowserCMD.R` implements the full functionality and allows to carry out the various enrichment methods and to produce HTML reports for interactive exploration of results.

The `inst/scripts` folder also contains a `README` file that comprehensively documents the usage of both scripts.

A A primer on terminology and statistical theory

A.1 Where does it all come from?

Test whether known biological functions or processes are over-represented (= enriched) in an experimentally-derived gene list, e.g. a list of differentially expressed (DE) genes. See [**?**] for a critical review.

Example: Transcriptomic study, in which 12,671 genes have been tested for differential expression between two sample conditions and 529 genes were found DE.

Among the DE genes, 28 are annotated to a specific functional gene set, which contains in total 170 genes. This setup corresponds to a 2×2 contingency table,

```
deTable <-
     matrix(c(28, 142, 501, 12000),
            nrow = 2,
            dimnames = list(c("DE", "Not.DE"),
                            c("In.gene.set", "Not.in.gene.set")))
deTable
```

```
##        In.gene.set Not.in.gene.set
## DE              28             501
## Not.DE         142           12000
```

where the overlap of 28 genes can be assessed based on the hypergeometric distribution. This corresponds to a one-sided version of Fisher's exact test, yielding here a significant enrichment.

```
fisher.test(deTable, alternative = "greater")
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  deTable
## p-value = 4.088e-10
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  3.226736      Inf
## sample estimates:
## odds ratio
##   4.721744
```

This basic principle is at the foundation of major public and commercial enrichment tools such as *DAVID* (https://david.ncifcrf.gov) and *Pathway Studio* (https://www.pathwaystudio.com).

Although gene set enrichment methods have been primarily developed and applied on transcriptomic data, they have recently been modified, extended and applied also in other fields of genomic and biomedical research. This includes novel approaches for functional enrichment analysis of proteomic and metabolomic data as well as genomic regions and disease phenotypes [**?**, **?**, **?**, **?**].

A.2 Gene sets, pathways & regulatory networks

Gene sets are simple lists of usually functionally related genes without further specification of relationships between genes.

Pathways can be interpreted as specific gene sets, typically representing a group of genes that work together in a biological process. Pathways are commonly divided in metabolic and signaling pathways. Metabolic pathways such as glycolysis represent biochemical substrate conversions by specific enzymes. Signaling pathways such as the MAPK signaling pathway describe signal transduction cascades from receptor proteins to transcription factors, resulting in activation or inhibition of specific target genes.

Gene regulatory networks describe the interplay and effects of regulatory factors (such as transcription factors and microRNAs) on the expression of their target genes.

A.3 Resources

GO (http://www.geneontology.org) and *KEGG* (http://www.genome.jp/kegg) annotations are most frequently used for the enrichment analysis of functional gene sets. Despite an increasing number of gene set and pathway databases, they are typically the first choice due to their long-standing curation and availability for a wide range of species.

The Gene Ontology (GO) consists of three major sub-ontologies that classify gene products according to molecular function (MF), biological process (BP) and cellular component (CC). Each ontology consists of GO terms that define MFs, BPs or CCs to which specific genes are annotated. The terms are organized in a directed acyclic graph, where edges between the terms represent relationships of different types. They relate the terms according to a parent-child scheme, i.e. parent terms denote more general entities, whereas child terms represent more specific entities.

The Kyoto Encyclopedia of Genes and Genomes (KEGG) is a collection of manually drawn pathway maps representing molecular interaction and reaction networks. These pathways cover a wide range of biochemical processes that can be divided in 7 broad categories: metabolism, genetic and environmental information processing, cellular processes, organismal systems, human diseases, and drug development. Metabolism and drug development pathways differ from pathways of the other 5 categories by illustrating reactions between chemical compounds. Pathways of the other 5 categories illustrate molecular interactions between genes and gene products.

A.4 Gene set analysis vs. gene set enrichment analysis

The two predominantly used enrichment methods are:

- Overrepresentation analysis (ORA), testing whether a gene set contains disproportional many genes of significant expression change, based on the procedure outlined in section **??**,
- Gene set enrichment analysis (GSEA), testing whether genes of a gene set accumulate at the top or bottom of the full gene vector ordered by direction and magnitude of expression change [**?**].

However, the term *gene set enrichment analysis* nowadays subsumes a general strategy implemented by a wide range of methods [**?**]. Those methods have in common the same goal, although approach and statistical model can vary substantially [**?**, **?**].

To better distinguish from the specific method, some authors use the term *gene set analysis* to denote the general strategy. However, there is also a specific method of this name [**?**].

A.5 Underlying null: competitive vs. self-contained

Goeman and Buehlmann, 2007, classified existing enrichment methods into *competitive* and *self-contained* based on the underlying null hypothesis [**?**].

- *Competitive* null hypothesis: the genes in the set of interest are at most as often DE as the genes not in the set,
- *Self-contained* null hypothesis: no genes in the set of interest are DE.

Although the authors argue that a self-contained null is closer to the actual question of interest, the vast majority of enrichment methods is competitive.

Goeman and Buehlmann further raise several critical issues concerning the 2×2 ORA:

- rather arbitrary classification of genes in DE / not DE,
- based on gene sampling, although sampling of subjects is appropriate,
- unrealistic independence assumption between genes, resulting in highly anti-conservative *p*-values.

With regard to these statistical concerns, GSEA is considered superior:

- takes all measured genes into account,
- subject sampling via permutation of class labels,
- the incorporated permutation procedure implicitly accounts for correlations between genes.

However, the simplicity and general applicability of ORA is unmet by subsequent methods improving on these issues. For instance, GSEA requires the expression data as input, which is not available for gene lists derived from other experiment types. On the other hand, the involved sample permutation procedure has been proven inaccurate and time-consuming [**?**, **?**, **?**].

A.6 Generations: ora, fcs & topology-based

Khatri *et al.*, 2012, have taken a slightly different approach by classifying methods along the timeline of development into three generations [**?**]:

1. Generation: ORA methods based on the 2×2 contingency table test,
2. Generation: functional class scoring (FCS) methods such as GSEA, which compute gene set (= functional class) scores by summarizing per-gene DE statistics,
3. Generation: topology-based methods, explicitly taking into account interactions between genes as defined in signaling pathways and gene regulatory networks ([**?**] for an example).

Although topology-based (also: network-based) methods appear to be most realistic, their straightforward application can be impaired by features that are not detectable on the transcriptional level (such as protein-protein interactions) and insufficient network knowledge [**?**, **?**].

Given the individual benefits and limitations of existing methods, cautious interpretation of results is required to derive valid conclusions. Whereas no single method is best suited for all application scenarios, applying multiple methods can be beneficial. This has been shown to filter out spurious hits of individual methods, thereby reducing the outcome to gene sets accumulating evidence from different methods [**?**, **?**].

B Frequently asked questions

1. **How to cite the *EnrichmentBrowser*?**

   Geistlinger L, Csaba G and Zimmer R. Bioconductor's EnrichmentBrowser: seamless navigation through combined results of set- & network-based enrichment analysis. *BMC Bioinformatics*, **17**:45, 2016.

2. **Is it possible to apply the *EnrichmentBrowser* to simple gene lists?**

   Enrichment methods implemented in the *EnrichmentBrowser* are, except for ORA, expression-based (and also draw their strength from that). The set-based methods GSEA, SAFE, and SAMGS use sample permutation, involving recomputation of differential expression, for gene set significance estimation, i.e. they require the complete expression matrix. The network-based methods require measures of differential expression such as fold change and p-value to score interactions of the network. In addition, visualization of enriched gene sets is explicitly designed for expression data. Thus, for simple gene list enrichment, tools like *DAVID* (https://david.ncifcrf.gov) and *GeneAnalytics* (http://geneanalytics.genecards.org) are more suitable, and it is recommended to use them for this purpose.