# Introduction to the *GSRI* package: Estimating Regulatory Effects utilizing the Gene Set Regulation Index

Julian Gehring, Clemens Kreutz, Kilian Bartholome, Jens Timmer

October 29, 2024

## 1 Introduction

The *GSRI* package estimates the number of significantly regulated genes in gene sets, assessing their differential effects between groups through statistical testing. The approach is based on the fact that p-values of a statistical test are uniformly distributed under the null hypothesis and shifted towards small values in the case of its violation. The resulting density distribution $\rho(p)$ of the p-values $p$ is then given as

$$\rho(p) = (1-r)\rho_0(p) + r\rho_A(p),$$

with the fraction $r$ of significant p-values, the uniform distribution $\rho_0(p)$ of the p-values under the null hypothesis, and the alternative distribution $\rho_A(p)$ of the p-values with a significant effect. In the cumulative density function (CDF) $F(p)$ this is equivalent to

$$F(p) = (1-r)p + r,$$

with the uniformly distributed $\rho_0(p)$ translating to a linear CDF with slope $1-r$ and intercept $r$. Through iterative fitting of this linear component, $r$ and thus the number of regulated genes can be estimated. An example for the probability and cumulative density distribution is shown in Figure 1.

The approach applied here does not require a cut-off value for the distinction between regulated and unregulated genes, nor any assumptions about the alternative distribution $\rho_A(p)$ of the p-values. Further, the method is independent of the statistical test used to assess the differential effect of genes in terms of p-values.

Estimates of the method include the number and fraction of regulated genes as well as the Gene Set Regulation Index $\eta$ (GSRI) for gene set. The GSRI $\eta$ is the 5% quantile of the distribution of the estimated number of differentially expressed genes obtained from bootstrapping the group samples. It indicates that with a probability of 95% more than $\eta$ genes in the gene set are differentially expressed. Utilizing the 5% quantile instead of the expectation $\hat{r}$ introduces a bias, but reduces the variability of the estimates and thereby improves the performance for a ranking of gene sets. The index can also be employed to test the hypothesis whether at least one gene in a gene set is regulated. Further,
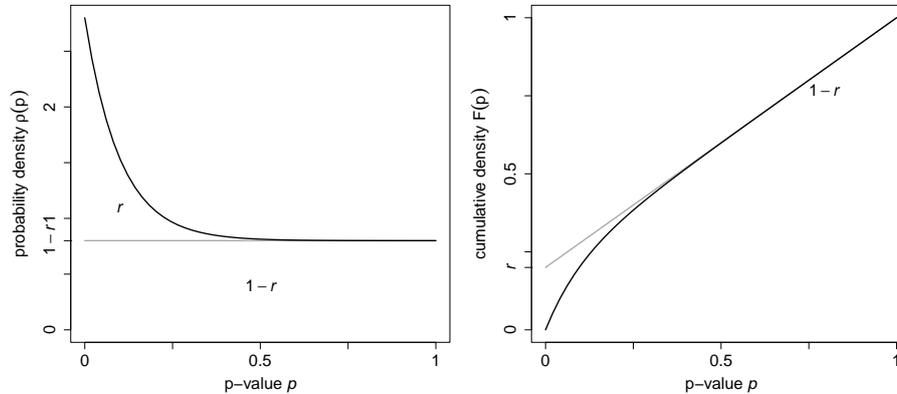
**Figure 1:** Distributions of p-values in the probability and cumulative density, as shown in the left and right panel, respectively. The ratio $r$ of significant tests have an unknown distribution shifted towards zero, while the remaining fraction of $1 - r$ tests exhibits a uniform distribution. This translates to a linear CDF with slope $1 - r$ and intercept $r$. By fitting the linear component of the CDF, as indicated by the dashed line, the ratio of significant tests can be estimated.

different gene sets can be compared or ranked according to the estimated amount of regulation. For details of the method, an application to experimental data, and a comparison with related approaches, see [1].

## 2 Data set

In this introduction we use the expression data set provided with the *Biobase* package. It contains the expression intensities of 26 microarray samples with a subset of 500 probe sets. The phenotypes associated with the samples are stored in the pheno data of the `ExpressionSet`, including the categorical variables *type* of disease and *sex* represented as factors, as well as the continuous *score* indicating the progress of the disease.

```
> library(Biobase)
> data(sample.ExpressionSet)
> eset <- sample.ExpressionSet
> eset

ExpressionSet (storageMode: lockedEnvironment)
assayData: 500 features, 26 samples
  element names: exprs, se.exprs
protocolData: none
phenoData
  sampleNames: A B ... Z (26 total)
  varLabels: sex type score
  varMetadata: labelDescription
```

2

```
featureData: none
experimentData: use 'experimentData(object)'
Annotation: hgu95av2

> exprs <- exprs(eset)
> phenotypes <- pData(phenoData(eset))
> summary(phenotypes)

     sex           type          score
 Female:11    Case   :15    Min.   :0.1000
 Male  :15    Control:11    1st Qu.:0.3275
                            Median :0.4150
                            Mean   :0.5369
                            3rd Qu.:0.7650
                            Max.   :0.9800
```

Please note that we are using this sample data to illustrate general workflows for the analysis of gene sets with the *GSRI* package. Therefore, the results obtained here should not be interpreted in the context of their biological meaning.

# 3  Analysis for a single gene set

Given the expression data we want to find out how many genes show a differential effect with respect to the phenotypic variables, in our case the groups *sex* and *type*. In a first step we include all genes in the analysis and focus on the *type* phenotype.

```
> library(GSRI)
> gAllProbes <- gsri(eset, phenotypes$type)
> gAllProbes

  pRegGenes pRegGenesSd nRegGenes GSRI(5%) nGenes
1 0.1425405   0.1428812        71        0    500
```

This indicates that around 14% of the genes seem to be regulated. However, taking the corresponding standard deviation of around 14% and the GSRI of 0% at the 5% confidence level into account, we have just an indication for a differential effect.

In the next step we exclude the controls of the Affymetrix microarray since they do not contain relevant information for our analysis. For this we define an object `gsAllGenes` of the class *GeneSet* with the subset of genes of interest. Note that in this case we could also use a subset of *eset* or *exprs* without an additional *GeneSet* object. For more details on how to define, import, and manipulate gene sets, please refer to the documentation of the *GSEABase* package [2].

```
> library(GSEABase)
> gs <- GeneSet(eset, setName="allGenes")
> ind <- grep("^AFFX", geneIds(gs), invert=TRUE)
> gsAllGenes <- gs[ind]
> gsAllGenes
```

```
setName: allGenes
geneIds: 31307_at, 31308_at, ..., 31739_at (total: 433)
geneIdType: Annotation (hgu95av2)
collectionType: ExpressionSet
details: use 'details(object)'

> gAllGenesType <- gsri(eset, phenotypes$type, gsAllGenes, name="allGenesType")
> gAllGenesSex <- gsri(exprs, phenotypes$sex, gsAllGenes, name="allGenesSex")
> gAllGenesType

            pRegGenes pRegGenesSd nRegGenes GSRI(5%) nGenes
allGenesType 0.1626202  0.1386983       70        0    433

> gAllGenesSex

           pRegGenes pRegGenesSd nRegGenes GSRI(5%) nGenes
allGenesSex        0   0.1101468        0        0    433
```

Taking only probes for human genes into acount we explore the effect of the *type* and *sex* variable. While the type of disease seems to have a differential effect on the gene expression, the sex of the patient shows no indication to play a role in this example.

The *GSEABase* package provides methods for importing gene sets from different sources. Here we import a gene set from an .xml file, with genes located on chromosome 17.

```
> gsChr17 <- getBroadSets(system.file("extdata", "c1chr17.xml", package="GSRI"))
> gsChr17

GeneSetCollection
  names: chr17 (1 total)
  unique identifiers: 32120_at, 40008_at, ..., 36103_at (526 total)
  types in collection:
    geneIdType: SymbolIdentifier (1 total)
    collectionType: BroadCollection (1 total)

> gChr17 <- gsri(eset, phenotypes$type, gsChr17)
> gChr17

      pRegGenes pRegGenesSd nRegGenes   GSRI(5%) nGenes
chr17 0.2398971   0.1387606         2 0.02437104     12
```

# 4 Analysis for multiple gene sets

It is often desirable to perform the GSRI analysis for an experimental data set, comparing several gene sets. This task can be approached with an object of the class *GeneSetCollection* combining multiple *GeneSet* objects.

We import five gene sets from a .gmt file and perform the analysis for those with respect to the *type* variable. Afterwards, we sort the gene sets according to the estimated number and fraction of genes, and export the results as a table to disk. The *summary* method provides a more detailed overview including the parameters used for the analysis.

```
> gmt <- getGmt(system.file("extdata", "c1c10.gmt", package="GSRI"))
> gCol5 <- gsri(eset, phenotypes$type, gmt)
> gCol5

        pRegGenes pRegGenesSd nRegGenes   GSRI(5%) nGenes
chr2     0.0000000   0.1451871         0 0.00000000     24
chr19q13 0.4010159   0.1746595         4 0.09083227     12
chr6     0.4679782   0.1312175        10 0.24046927     23
chr8     0.1126934   0.1742302         1 0.00000000     12
chr9     0.3778584   0.1491297         3 0.13259385     10

> gCol5Sort <- sortGsri(gCol5, c("nRegGenes", "pRegGenes"))
> summary(gCol5Sort)

* Summary of the GSRI analysis for 5 gene sets:

** Results
         pRegGenes pRegGenesSd nRegGenes GSRI(5%) nGenes
chr6        0.4680      0.1312        10   0.2405     23
chr19q13    0.4010      0.1747         4   0.0908     12
chr9        0.3779      0.1491         3   0.1326     10
chr8        0.1127      0.1742         1   0.0000     12
chr2        0.0000      0.1452         0   0.0000     24

** Parameter
 nBoot alpha grenander weighting
   100  0.05      TRUE     FALSE

> exportFile <- tempfile()
> export(gCol5Sort, exportFile)
```

# 5   Adaption of statistical tests

As pointed out in the introduction, the GSRI approach is independent of the
underlying statistical test. By default a t-test is used to assess the differential
effect between two groups. With an F-test an arbitrary number of groups can
be used for the analysis, while for two groups it is equivalent to the t-test.

As an example we arbitrarily define three groups based on the score variable
indicating the progress of the disease. For this analysis we use the F-test rowF
provided with this package.

```
> phenotypes$class <- cut(phenotypes$score, seq(0, 1, length.out=4), label=c("low", "mediu
> summary(phenotypes$class)

   low medium   high
     7      8     11

> g3 <- gsri(eset, phenotypes$class, gsChr17, test=rowF)
> g3

       pRegGenes pRegGenesSd nRegGenes GSRI(5%) nGenes
chr17 0.07862739   0.1329572         0        0     12
```

The GSRI approach has several parameters that can be changed in order to adapt the analysis. For illustration we rename the gene set, change the number of bootstraps and confidence level for the GSRI calculation, and use a classical ECDF instead of the modified Grenander estimator for the cumulative density.

```
> g3arg2 <- gsri(eset, phenotypes$class, gsChr17, test=rowF, name="chr17_2", nBoot=50, alp
> g3arg2

        pRegGenes pRegGenesSd nRegGenes GSRI(10%) nGenes
chr17_2 0.1055541   0.1167119         1         0     12
```

We can also easily implement our own statistical tests for the GSRI analysis. Next, we want to apply an approach taken by the *limma* package [3] which as an increased power for small sample sizes. The canonical structure of the test function has to be called as function(exprs, groups, id, index, testArgs), with *exprs* the matrix of expression intensities, *groups* the factor of groups defining the differential contrast, *id* the indices for the genes part of the current gene set, *index* the indices for the samples in the bootstrapping, and *testArgs* the list with optional arguments used by the test function.

```
> library(limma)
> limmaTest <- function(exprs, groups, id, index, testArgs) {
+     design <- cbind(offset=1, diff=groups)
+     fit <- lmFit(exprs[ ,index], design)
+     fit <- eBayes(fit)
+     pval <- fit$p.value[id,"diff"]
+     return(pval)
+ }
> g3Limma <- gsri(eset, phenotypes$type, gsChr17, test=limmaTest)
> g3Limma

      pRegGenes pRegGenesSd nRegGenes   GSRI(5%) nGenes
chr17 0.2397355   0.1506783         2 0.005084557     12
```

# 6  Visualization

The results of the GSRI analysis can be visualized, showing the empirical cumulative p-values distribution along with the fit of the null distribution $\rho_0(p)$ as well as the estimated fraction $\hat{r}$ of significant genes and the GSRI $\eta$.

```
> plot(g3)
```

The *plot* method has an advanced system in order to customize the plot in different aspects. This allows us to directly adapt nearly any property of the figure. For a detailed description, please refer to the documentation of the *plot* method.

```
> plot(gCol5, 5, ecdf=list(type="o"), plot=list(xlab="p", main="GSRI results: chr9"), reg=
```
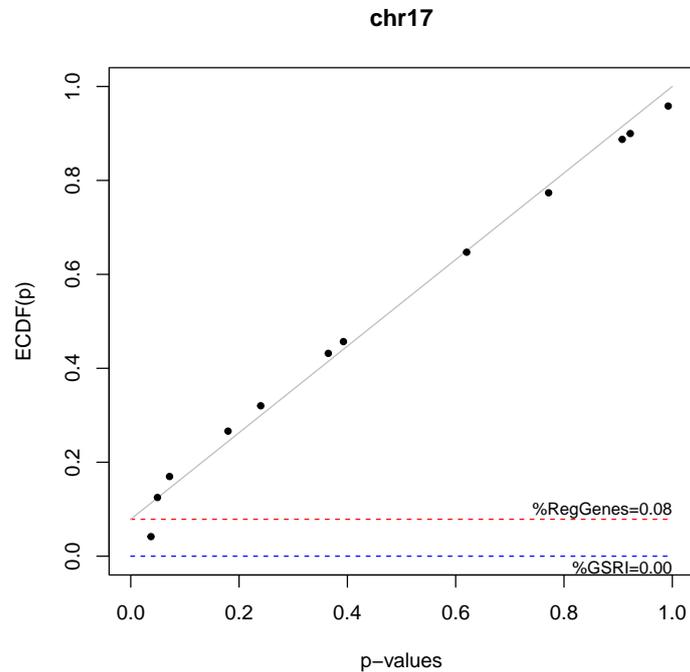
**Figure 2:** Visualization of GSRI results

# 7 Weighting of genes in gene sets

In contrast to other approaches estimating the degree of regulation, the *GSRI* package does also allow assign the weighting of each gene in the calculation. Such a step is useful for including additional information in the estimation process, for example the certainty that a gene is part of a gene set.

In the following we use a very simple approach in defining weights for the gene sets based on the Gene Ontology (GO) annotation. For genes with experimental evidence, we assign higher weights than for those without. Please note that the weights used here are defined arbitrarily and more sophisticated approaches can be used in the actual analysis.

```
> library(hgu95av2.db)
> gNames <- rownames(exprs(eset))
> ind <- Lkeys(hgu95av2GO) %in% gNames
> evidence <- factor(toTable(hgu95av2GO)[ind,"Evidence"])
> summary(evidence)

 EXP  HDA  HEP  HMP  HTP  IBA   IC  IDA  IEA  IEP  IGI  IMP  IPI
  35  285    3    4   20 1883   49 3007 2604   27  135  834  693
 ISA  ISO  ISS  NAS   ND  RCA  TAS
  64    8 1157  459   25    9 1199

> l <- lapply(gNames, function(name, names, evidence) evidence[names %in% name], gNames, e
> expInd <- sapply(l, function(l) any(l %in% "EXP"))
```

7
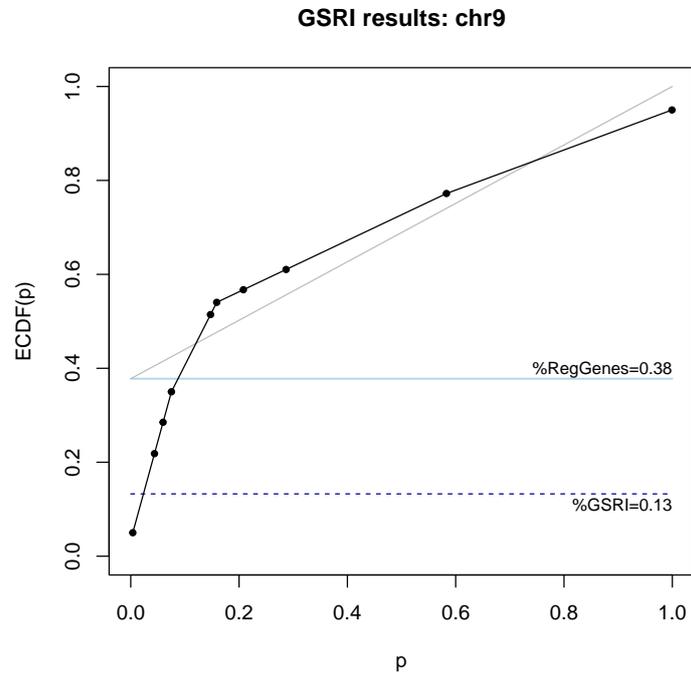
**GSRI results: chr9**



**Figure 3:** Visualization of GSRI results, with customized parameters.

```
> goWeight <- rep(0.5, length.out=length(expInd))
> goWeight[expInd] <- 1
> gCol5go <- gsri(eset, phenotypes$type, weight=goWeight)
> gCol5go

  pRegGenes pRegGenesSd nRegGenes GSRI(5%) nGenes
1 0.1407464   0.1412441        70        0    500

> gCol5go2 <- gsri(eset, phenotypes$type, gmt, weight=goWeight)
> gCol5go2

             pRegGenes pRegGenesSd nRegGenes    GSRI(5%) nGenes
chr2      2.220446e-16   0.1324177         0  0.00000000     24
chr19q13  3.062520e-01   0.2094432         3  0.04558399     12
chr6      4.199143e-01   0.1367127         9  0.20617498     23
chr8      9.528374e-02   0.1506949         1  0.00000000     12
chr9      3.778584e-01   0.1621898         3  0.08867302     10
```

8

# References

[1] Kilian Bartholome, Clemens Kreutz, and Jens Timmer. Estimation of gene induction enables a relevance-based ranking of gene sets. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, 16(7):959–967, July 2009. PMID: 19580524.

[2] Martin Morgan, Seth Falcon, and Robert Gentleman. *GSEABase: Gene set enrichment data structures and methods.*

[3] Gordon K. Smyth. Limma: linear models for microarray data. In R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, and W. Huber, editors, *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, pages 397–420. Springer, New York, 2005.

# Session info

- R version 4.4.1 (2024-06-14), `x86_64-pc-linux-gnu`

- Running under: `Ubuntu 24.04.1 LTS`

- Matrix products: default

- BLAS: `/home/biocbuild/bbs-3.20-bioc/R/lib/libRblas.so`

- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0`

- Base packages: base, datasets, grDevices, graphics, methods, stats, stats4, utils

- Other packages: AnnotationDbi 1.68.0, Biobase 2.66.0, BiocGenerics 0.52.0, GSEABase 1.68.0, GSRI 2.54.0, IRanges 2.40.0, S4Vectors 0.44.0, XML 3.99-0.17, annotate 1.84.0, fdrtool 1.2.18, graph 1.84.0, hgu95av2.db 3.13.0, limma 3.62.0, org.Hs.eg.db 3.20.0

- Loaded via a namespace (and not attached): Biostrings 2.74.0, DBI 1.2.3, GenomeInfoDb 1.42.0, GenomeInfoDbData 1.2.13, KEGGREST 1.46.0, KernSmooth 2.23-24, Matrix 1.7-1, MatrixGenerics 1.18.0, R6 2.5.1, RColorBrewer 1.1-3, RSQLite 2.3.7, UCSC.utils 1.2.0, XVector 0.46.0, bit 4.5.0, bit64 4.5.2, bitops 1.0-9, blob 1.2.4, boot 1.3-31, caTools 1.18.3, cachem 1.1.0, cli 3.6.3, compiler 4.4.1, crayon 1.5.3, fastmap 1.2.0, genefilter 1.88.0, gplots 3.2.0, grid 4.4.1, gtools 3.9.5, httr 1.4.7, jsonlite 1.8.9, lattice 0.22-6, les 1.56.0, matrixStats 1.4.1, memoise 2.0.1, pkgconfig 2.0.3, png 0.1-8, rlang 1.1.4, splines 4.4.1, statmod 1.5.0, survival 3.7-0, tools 4.4.1, vctrs 0.6.5, xtable 1.8-4, zlibbioc 1.52.0