Gene Set Regulation Index (GSRI)

Julian Gehring, Clemens Kreutz, Kilian Bartholomé

February 16, 2011

Abstract

This document gives an introduction in how to use the *GSRI* package. It estimates the number of significantly regulated genes in gene sets using the Gene Set Regulation Index (GSRI).

1 Introduction

The *GSRI* package calculates the number of differentially expressed genes in a gene set. The method does not require a cut-off value for the distinction between regulated and unregulated genes. This approach is based on the fact that p-values obtained from a statistical test are uniformly distributed under the null hypothesis and accumulated around zero for the alternative hypothesis. Estimates of the method include the number and percentage of regulated genes as well as the corresponding standard errors, and the Gene Set Regulation Index (GSRI). The GSRI 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 GSRI genes in the gene set are differentially expressed. Utilizing the 5% quantile instead of the expectation introduces a bias, but improves the precision, i.e. reduces the variability of the estimates and thereby improves the performance for a ranking of gene sets. This index can also be employed to test the hypothesis whether at least one gene in a set is regulated and to compare and rank the regulation of different gene sets. For details on the method, application to biological data sets and comparison to existing methods, see [1].

2 Calculate GSRI for a single gene set

First have a look at how to compute the GSRI for a single gene set.

Simulate a gene set with $n = 100$ genes with 30 regulated ones and $m = 40$ samples (20 controls and 20 treatments). The expression data is stored in `expdata`, the phenotypes of the samples in `phenotype`.

```
> set.seed(0)
> data <- matrix(rnorm(4000, mean = 0), nrow = 100,
+     ncol = 40)
> data[1:30, 1:20] <- rnorm(600, mean = 1)
> phenotype <- factor(rep(c(0, 1), each = 20))
> geneSetName <- "Test Gene Set"
```

Calculate the gene set regulation index. Plot the results and omit writing the results to disc.

```
> library(GSRI)
> res <- gsri(data, phenotype, geneSetName, plotResults = TRUE,
+     writeResults = FALSE)
```
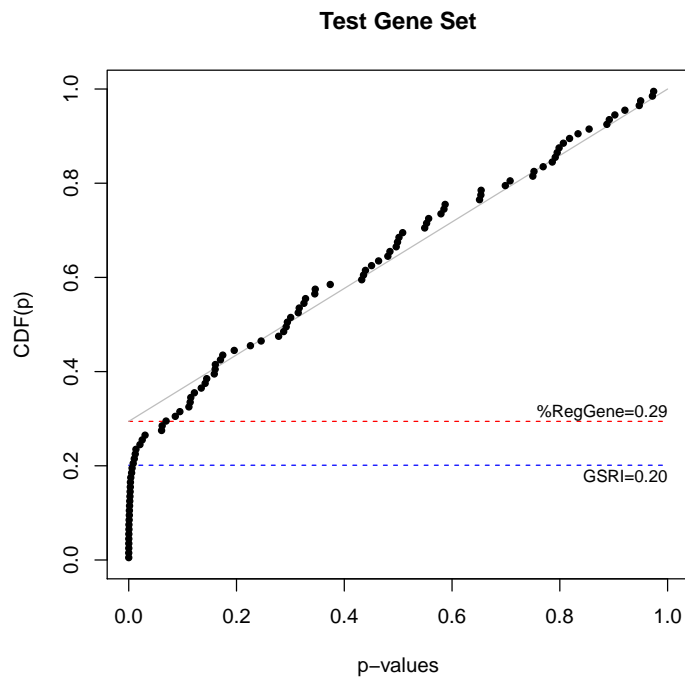
Test Gene Set

Figure 1: Result figure from the `gsri` function. The plot shows the cumulative distribution of the p-values of the gene set. The black line indicates the estimate for the uniform distribution consistent with the null hypothesis. The estimated percentage of regulated genes and the GSRI are indicated in red and green.

```
> print(res)

$geneSet
[1] "Test Gene Set"

$percRegGenes
[1] 0.2943323

$percRegGenesSd
[1] 0.05640529

$numRegGenes
[1] 29.43323
```

```
$numRegGenesSd
[1] 5.640529

$gsri
[1] 0.2011746

$nGenes
[1] 100
```

In this case the analysis reports 29 regulated genes and a GSRI of 0.2.

You can also include the Grenander correction for the distribution of p-values from the *fdrtool* package. It uses the assumption about the concave shape of the cumulative density distribution. Using the Grenander estimate reduces the variance, i.e. makes the approach more stable especially for small gene sets. On the downside the number of significant genes is overestimated for few regulated genes.

```
> res2 <- gsri(data, phenotype, "Test Gene Set with Grenander",
+     useGrenander = TRUE)
> res2$numRegGenes

[1] 28.5095
```
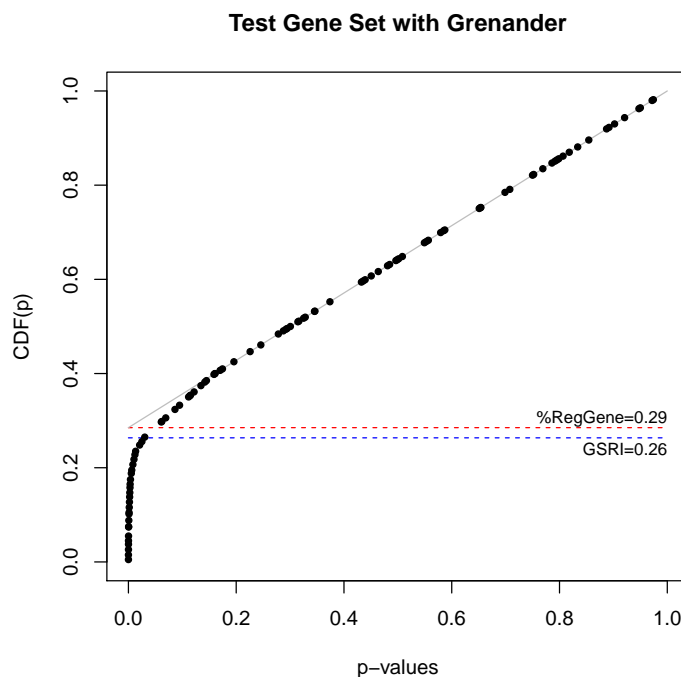

3 Apply own statistical tests

In the example shown above a t-test was used as a statistical test to compute the p-values. Groups were defined by `phenotype`. In the GSRI package itself

two statistical tests are implemented. It provides a t-test and an F-test from the *genefilter* package, with the t-test as default. These tests can be chosen with the `test` argument, with character strings `"test"` and `"ftest"` respectively.

Depending on your experimental design it may be appropriate to choose other tests. These can be easily used with the *GSRI* package.

Your function has to accept the inputs `data` (as matrix) and `phenotype` (as factor) like they were passed to the `gsri` routine. The optional argument `testArgs` can contain additional parameters. It will only be passed to your test function if it is specified. The function has to return a vector of p-values, one for each gene.

An example for a user defined statistical test is shown here with the `lm` lm function from the *stats* package. The `statFcn` function calculates a p-value for a single row of the data set, i.e. for a single gene.

```
> statFcn <- function(d, p) {
+     m <- lm(d ~ p)
+     pval <- summary(m)$coefficients[2, 4]
+ }
```

The user defined test function has to apply a test statistics on the whole data set. The following wrapper function can be used as a templete for test functions accepting only data vectors as an input. The `apply` function implements the loop over all genes.

```
> testFcn <- function(data, phenotype) {
+     pvals <- apply(data, 1, statFcn, phenotype)
+     return(pvals)
+ }
```

The function`testFcn` is simply passed to the `gsri` routine.

```
> res3 <- gsri(data, phenotype, geneSetName, test = testFcn,
+     plotResults = FALSE, nBootstraps = 20)
```

4   Read data from files for multiple gene sets

With the `gsriFromFile` function all data can be directly read from disk without prior importing and compute results for a number of gene sets. In this example `data.gct` contains the expression data, `phenotype.cls` the phenotypes and `geneSets.gmt` a list of possible gene sets. Calculation will be run for all specified gene sets.

The files are located in the same directory as this vignette. The `system.file` function is used to construct valid paths independent of your current working directory. Please note that this is not mandatory for your own analysis.

These files have been created for demonstration and do not represent real biological data. Also note that real data will contain more genes and larger gene sets. To get more information on the file formats you may have a look at these files located in the same directory as this vignette or at `http://www.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats`. To obtain experimental data sets, see e.g. `http://www.broadinstitute.org/gsea/`.

The four annotated pathways contain 7, 0, 4 and 2 regulated genes. Compare this with the estimated number of regulated genes.

```
> dataFileName <- system.file("extdata", "data.gct",
+     package = "GSRI")
> phenotypeFileName <- system.file("extdata", "phenotype.cls",
+     package = "GSRI")
> geneSetFileName <- system.file("extdata", "geneSets.gmt",
+     package = "GSRI")
> res4 <- gsriFromFile(dataFileName, phenotypeFileName,
+     geneSetFileName)
> res4$numRegGenes

[1] 6.600 0.000 3.810 1.212
```

References

[1] Kilian Bartholomé, 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.

Session info

- R version 2.12.1 (2010-12-16), `x86_64-pc-linux-gnu`

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

- Other packages: GSRI 1.3.2, fdrtool 1.2.6

- Loaded via a namespace (and not attached): AnnotationDbi 1.13.13, Biobase 2.11.8, DBI 0.2-5, RSQLite 0.9-4, annotate 1.29.2, boot 1.2-43, genefilter 1.33.0, splines 2.12.1, survival 2.36-2, tools 2.12.1, xtable 1.5-6