GSVA: The Gene Set Variation Analysis package for microarray and RNA-seq data

Sonja Hänzelmann[1], Robert Castelo[1] and Justin Guinney[2]

February 1, 2017

1. Research Program on Biomedical Informatics (GRIB), Hospital del Mar Research Institute (IMIM) and Universitat Pompeu Fabra, Parc de Recerca Biomèdica de Barcelona, Doctor Aiguader 88, 08003 Barcelona, Catalonia, Spain

2. Sage Bionetworks, 1100 Fairview Ave N., Seattle, Washington, 98109 USA

Abstract

The GSVA package implements a non-parametric unsupervised method, called Gene Set Variation Analysis (GSVA), for assessing gene set enrichment (GSE) in gene expression microarray and RNA-seq data. In contrast to most GSE methods, GSVA performs a change in coordinate systems, transforming the data from a gene by sample matrix to a gene set by sample matrix. Thereby allowing for the evaluation of pathway enrichment for each sample. This transformation is done without the use of a phenotype, thus facilitating very powerful and open-ended analyses in a now pathway centric manner. In this vignette we illustrate how to use the GSVA package to perform some of these analyses using published microarray and RNA-seq data already pre-processed and stored in the companion experimental data package GSVAdata.

1 Introduction

Gene set enrichment analysis (GSEA) (see **??**) is a method designed to assess the concerted behavior of functionally related genes forming a set, between two well-defined groups of samples. Because it does not rely on a "gene list" of interest but on the entire ranking of genes, GSEA has been shown to provide greater sensitivity to find gene expression changes of small magnitude that operate coordinately in specific sets of functionally related genes. However, due to the reduced costs in genome-wide gene-expression assays, data is being produced under more complex experimental designs that involve multiple RNA sources enriched with a wide spectrum of phenotypic and/or clinical information. The Cancer Genome Atlas (TCGA) project (see `http://cancergenome.nih.gov`) and the data deposited on it constitute a canonical example of this situation.

To facilitate the functional enrichment analysis of this kind of data, we developed Gene Set Variation Analysis (GSVA) which allows the assessment of the underlying pathway activity variation by transforming the gene by sample matrix into a gene set by sample matrix without the *a priori* knowledge of the experimental design. The method is both non-parametric and unsupervised, and bypasses the conventional approach of explicitly modeling phenotypes within enrichment scoring algorithms. Focus is therefore placed on the *relative* enrichment of pathways across the sample space rather than the *absolute* enrichment with respect to a phenotype. The value of this approach is that it permits the use of traditional analytical methods such as classification, survival analysis, clustering, and correlation analysis in a pathway focused manner. It also facilitates sample-wise comparisons between pathways and other complex data types such as microRNA expression or binding data, copy-number variation (CNV) data, or single nucleotide polymorphisms (SNPs). However, for case-control experiments, or data with a moderate to small sample size (< 30), other GSE methods that explicitly include the phenotype in their model are more likely to provide greater statistical power to detect functional enrichment.

In the rest of this vignette we describe briefly the methodology behind GSVA, give an overview of the functions implemented in the package and show a few applications. The interested reader is referred to (**?**) for more comprehensive explanations and more complete data analysis examples with GSVA, as well as for citing GSVA if you use it in your own work.

2 GSVA enrichment scores

A schematic overview of the GSVA method is provided in Figure **??**, which shows the two main required inputs: a matrix $X = \{x_{ij}\}_{p \times n}$ of normalized expression values (see Methods for details on the pre-processing steps) for p genes by n samples, where typically $p \gg n$, and a collection of gene sets $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$. We shall denote by x_i the expression profile of the i-th gene, by x_{ij} the specific expression value of the i-th gene in the j-th sample, and by γ_k the subset of row indices in X such that $\gamma_k \subset \{1, \ldots p\}$ defines a set of genes forming a pathway or some other functional unit. Let $|\gamma_k|$ be the number of genes in γ_k.

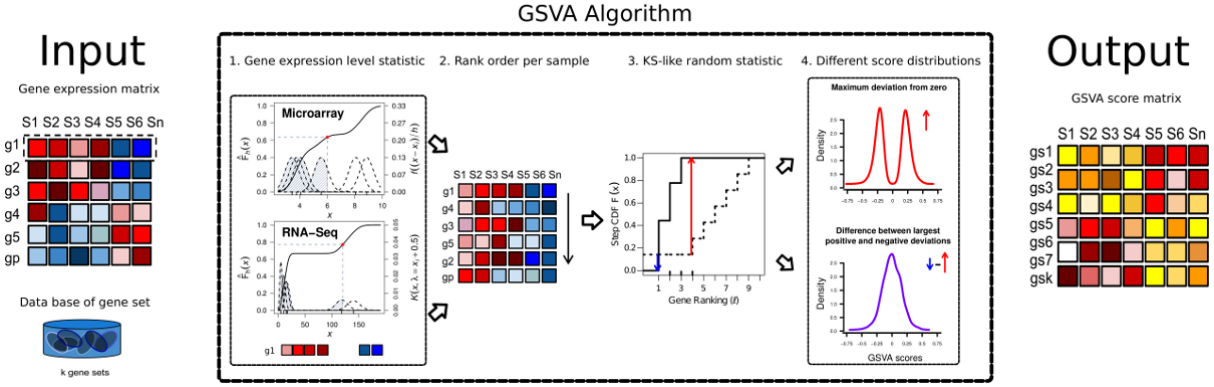


Figure 1: **GSVA methods outline.** The input for the GSVA algorithm are a gene expression matrix in the form of log2 microarray expression values or RNA-seq counts and a database of gene sets. 1. Kernel estimation of the cumulative density function (kcdf). The two plots show two simulated expression profiles mimicking 6 samples from microarray and RNA-seq. The x-axis corresponds to expression values where each gene is lowly expressed in the four samples with lower values and highly expressed in the other two. The scale of the kcdf is on the left y-axis and the scale of the Gaussian and Poisson kernels is on the right y-axis. 2. The expression-level statistic is rank ordered for each sample. 3. For every gene set, the Kolmogorov-Smirnov-like rank statistic is calculated. The plot illustrates a gene set consisting of 3 genes out of a total number of 10 with the sample-wise calculation of genes inside and outside of the gene set. 4. The GSVA enrichment score is either the difference between the two sums or the maximum deviation from zero. The two plots show two simulations of the resulting scores under the null hypothesis of no gene expression change (see main text). The output of the algorithm is matrix containing pathway enrichment profiles for each gene set and sample.

GSVA starts by evaluating whether a gene i is highly or lowly expressed in sample j in the context of the sample population distribution. Probe effects can alter hybridization intensities in microarray data such that expression values can greatly differ between two non-expressed genes**?**. Analogous gene-specific biases, such as GC content or gene length have been described in RNA-seq data**?**. To bring distinct expression profiles to a common scale, an expression-level statistic is calculated as follows. For each gene expression profile $x_i = \{x_{i1}, \ldots, x_{in}\}$, a non-parametric kernel estimation of its cumulative density function is performed using a Gaussian kernel (**?**, pg. 148) in the case of microarray data:

$$\hat{F}_{h_i}(x_{ij}) = \frac{1}{n} \sum_{k=1}^{n} \int_{-\infty}^{\frac{x_{ij}-x_{ik}}{h_i}} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt \,, \tag{1}$$

where h_i is the gene-specific bandwidth parameter that controls the resolution of the kernel estimation, which is set to $h_i = s_i/4$, where s_i is the sample standard deviation of the i-th gene (Figure **??**, step 1). In the case of RNA-seq data, a discrete Poisson kernel **?** is employed:

$$\hat{F}_r(x_{ij}) = \frac{1}{n} \sum_{k=1}^{n} \sum_{y=0}^{x_{ij}} \frac{e^{-(x_{ik}+r)}(x_{ik}+r)^y}{y!} \,, \tag{2}$$

where $r = 0.5$ in order to set the mode of the Poisson kernel at each x_{ik}, because the mode of a Poisson

distribution with an integer mean λ occurs at λ and $\lambda - 1$ and at the largest integer smaller than λ when λ is continuous.

Let z_{ij} denote the previous expression-level statistic $\hat{F}_{h_i}(x_{ij})$, or $\hat{F}_r(x_{ij})$, depending on whether x_{ij} are continuous microarray, or discrete count RNA-seq values, respectively. The following step condenses expression-level statistics into gene sets by calculating sample-wise enrichment scores. To reduce the influence of potential outliers, we first convert z_{ij} to ranks $z_{(i)j}$ for each sample j and normalize further $r_{ij} = |p/2 - z_{(i)j}|$ to make the ranks symmetric around zero (Figure **??**, step 2). This is done to up-weight the two tails of the rank distribution when computing the final enrichment score.

We assess the enrichment score similar to the GSEA and ASSESS methods **??** using the Kolmogorov-Smirnov (KS) like random walk statistic (Figure **??**, step 3):

$$\nu_{jk}(\ell) = \frac{\sum_{i=1}^{\ell} |r_{ij}|^{\tau} I(g_{(i)} \in \gamma_k)}{\sum_{i=1}^{p} |r_{ij}|^{\tau} I(g_{(i)} \in \gamma_k)} - \frac{\sum_{i=1}^{\ell} I(g_{(i)} \notin \gamma_k)}{p - |\gamma_k|}, \tag{3}$$

where τ is a parameter describing the weight of the tail in the random walk (default $\tau = 1$), γ_k is the k-th gene set, $I(g_{(i)} \in \gamma_k)$ is the indicator function on whether the i-th gene (the gene corresponding to the i-th ranked expression-level statistic) belongs to gene set γ_k, $|\gamma_k|$ is the number of genes in the k-th gene set, and p is the number of genes in the data set. Conceptually, Eq. **??** produces a distribution over the genes to assess if the genes in the gene set are more likely to be found at either tail of the rank distribution (see **??** for a more detailed description).

We offer two approaches for turning the KS like random walk statistic into an enrichment statistic (ES) (also called GSVA score), the classical maximum deviation method **???** and a normalized ES. The first ES is the maximum deviation from zero of the random walk of the j-th sample with respect to the k-th gene set :

$$ES_{jk}^{\max} = \nu_{jk}[\arg \max_{\ell=1,\ldots,p} |\nu_{jk}(\ell)|]. \tag{4}$$

For each gene set k, this approach produces a distribution of enrichment scores that is bimodal (Figure **??**, step 4, top panel). This is an intrinsic property of the KS like random walk, which generates non-zero maximum deviations under the null distribution. In GSEA **?** it is also observed that the empirical null distribution obtained by permuting phenotypes is bimodal and, for this reason, significance is determined independently using the positive and negative sides of the null distribution. In our case, we would like to provide a standard Gaussian distribution of enrichment scores under the null hypothesis of no change in pathway activity throughout the sample population. For this purpose we propose a second, alternative score that produces an ES distribution approximating this requirement (Figure **??**, step 4, bottom panel):

$$ES_{jk}^{\text{diff}} = \left|ES_{jk}^{+}\right| - \left|ES_{jk}^{-}\right| = \max_{\ell=1,\ldots,p}(0, \nu_{jk}(\ell)) - \min_{\ell=1,\ldots,p}(0, \nu_{jk}(\ell))\,, \tag{5}$$

where ES_{jk}^{+} and ES_{jk}^{-} are the largest positive and negative random walk deviations from zero, respectively, for sample j and gene set k. This statistic may be compared to the Kuiper test statistic **?**, which sums the maximum and minimum deviations to make the test statistic more sensitive in the tails. In contrast, our test statistic penalizes deviations that are large in both tails, and provides a "normalization" of the enrichment score by subtracting potential noise. There is a clear biological interpretation of this statistic, it emphasizes genes in pathways that are concordantly activated in one direction only, either over-expressed or under-expressed relative to the overall population. For pathways containing genes strongly acting in both directions, the deviations will cancel each other out and show little or no enrichment. Because this statistic is unimodal and approximately normal, downstream analyses which may impose distributional assumptions on the data are possible.

Figure **??**, step 4 shows a simple simulation where standard Gaussian deviates are independenty sampled from $p = 20,000$ genes and $n = 30$ samples, thus mimicking a null distribution of no change in gene expression. One hundred gene sets are uniformly sampled at random from the p genes with sizes ranging from 10 to 100 genes. Using these two inputs, we calculate the maximum deviation ES and the normalized ES. The resulting distributions are depicted in Figure **??**, step 4 and in the larger figure below, illustrating the previous description.

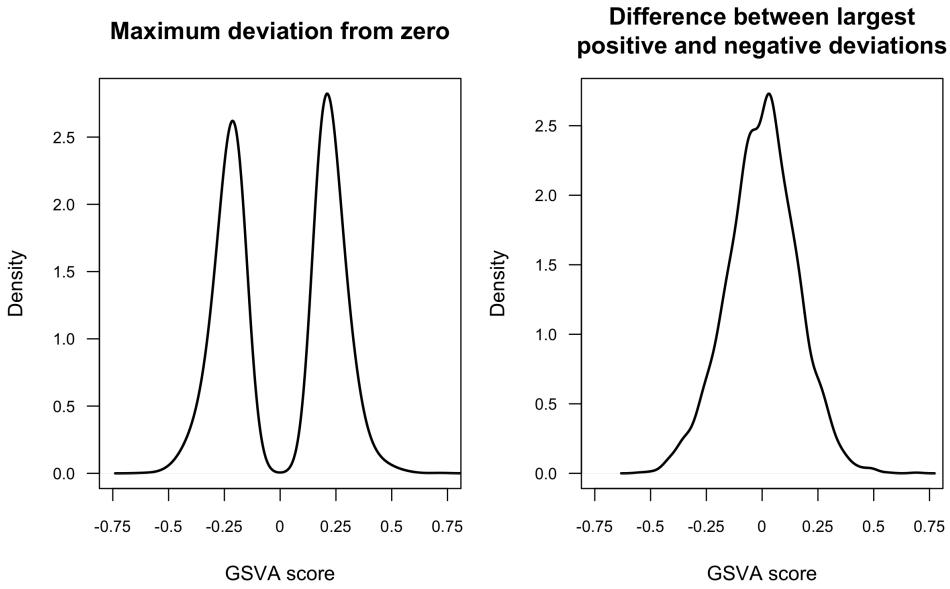
```
> library(GSVA)
> p <- 20000    ## number of genes
```

```
> n <- 30       ## number of samples
> nGS <- 100    ## number of gene sets
> min.sz <- 10  ## minimum gene set size
> max.sz <- 100 ## maximum gene set size
> X <- matrix(rnorm(p*n), nrow=p, dimnames=list(1:p, 1:n))
> dim(X)

[1] 20000    30

> gs <- as.list(sample(min.sz:max.sz, size=nGS, replace=TRUE)) ## sample gene set sizes
> gs <- lapply(gs, function(n, p) sample(1:p, size=n, replace=FALSE), p) ## sample gene sets
> es.max <- gsva(X, gs, mx.diff=FALSE, verbose=FALSE, parallel.sz=1)$es.obs
> es.dif <- gsva(X, gs, mx.diff=TRUE, verbose=FALSE, parallel.sz=1)$es.obs

> par(mfrow=c(1,2), mar=c(4, 4, 4, 1))
> plot(density(as.vector(es.max)), main="Maximum deviation from zero",
+      xlab="GSVA score", lwd=2, las=1, xaxt="n", xlim=c(-0.75, 0.75), cex.axis=0.8)
> axis(1, at=seq(-0.75, 0.75, by=0.25), labels=seq(-0.75, 0.75, by=0.25), cex.axis=0.8)
> plot(density(as.vector(es.dif)), main="Difference between largest\npositive and negative deviation
+      xlab="GSVA score", lwd=2, las=1, xaxt="n", xlim=c(-0.75, 0.75), cex.axis=0.8)
> axis(1, at=seq(-0.75, 0.75, by=0.25), labels=seq(-0.75, 0.75, by=0.25), cex.axis=0.8)
```



Although the GSVA algorithm itself does not evaluate statistical significance for the enrichment of gene sets, significance with respect to one or more phenotypes can be easily evaluated using conventional statistical models. Likewise, false discovery rates can be estimated by permuting the sample labels (Methods). Examples of these techniques are provided in the following section.

3 Overview of the package

The GSVA package implements the methodology described in the previous section in the function `gsva()` which requires two main input arguments: the gene expression data and a collection of gene sets. The expression data can be provided either as a *matrix* object of genes (rows) by sample (columns) expression values, or as an *ExpressionSet* object. The collection of gene sets can be provided either as a *list* object with names identifying gene sets and each entry of the list containing the gene identifiers of the genes forming the corresponding set, or as a *GeneSetCollection* object as defined in the GSEABase package.

When the two main arguments are an *ExpressionSet* object and a *GeneSetCollection* object, the `gsva()` function will first translate the gene identifiers used in the *GeneSetCollection* object into the corresponding feature identifiers of the *ExpressionSet* object, according to its corresponding annotation package. This translation is carried out by an internal call to the function `mapIdentifiers()` from the GSEABase package. This means that both input arguments may specify features with different types of identifiers, such as Entrez IDs and probeset IDs, and the GSEABase package will take care of mapping them to each other.

A second filtering step is applied that removes genes without matching features in the *ExpressionSet* object. If the expression data is given as a *matrix* object then only the latter filtering step will be applied by the `gsva()` function and, therefore, it will be the responsibility of the user to have the same type of identifiers in both the expression data and the gene sets.

After these automatic filtering steps, we may additionally filter out gene sets that do not meet a minimum and/or maximum size specified by the optional arguments `min.sz` and `max.sz` which are set by default to 1 and `Inf`, respectively. Finally, the `gsva()` function will carry out the calculations specified in the previous section and return a gene set by sample matrix of GSVA enrichment scores in the form of a *matrix* object when this was the class of the input expression data object. Otherwise, it will return an *ExpressionSet* object inheriting all the corresponding phenotypic information from the input data.

An important argument of the `gsva()` function is the flag `mx.diff` which is set to `TRUE` by default. Under this default setting, GSVA enrichment scores are calculated using Equation **??**, and therefore, are more amenable by analysis techniques that assume the data to be normally distributed. When setting `mx.diff=FALSE`, then Equation **??** is employed, calculating enrichment in an analogous way to classical GSEA which typically provides a bimodal distribution of GSVA enrichment scores for each gene.

Since the calculations for each gene set are independent from each other, the `gsva()` function offers two possibilities to perform them in parallel. One consists of loading the library snow, which will enable the parallelization of the calculations through a cluster of computers. In order to activate this option we should specify in the argument `parallel.sz` the number of processors we want to use (default is zero which means no parallelization is going to be employed). The other is loading the library parallel and then the `gsva()` function will use the core processors of the computer where R is running. If we want to limit `gsva()` in the number of core processors that should be used, we can do it by specifying the number of cores in the `parallel.sz` argument.

The other two functions of the GSVA package are `filterGeneSets()` and `computeGeneSetsOverlaps()` that serve to explicitly filter gene sets by size and by pair-wise overlap, respectively. Note that the size filter can also be applied within the `gsva()` function call.

The `gsva()` function also offers the following three other unsupervised GSE methods that calculate single sample pathway summaries of expression and which can be selected through the `method` argument:

- `method="plage"` (**?**). Pathway level analysis of gene expression (PLAGE) standardizes first expression profiles into z-scores over the samples and then calculates the singular value decomposition $Z_\gamma = UDV'$ on the z-scores of the genes in the gene set. The coefficients of the first right-singular vector (first column of V) are taken as the gene set summaries of expression over the samples.

- `method="zscore"` (**?**). The combined z-score method also, as PLAGE, standardizes first expression profiles into z-scores over the samples, but combines them together for each gene set at each individual sample as follows. Given a gene set $\gamma = \{1, \ldots, k\}$ with z-scores $Z_1, \ldots, Z_k$ for each gene, the combined z-score Z_γ for the gene set γ is defined as:

$$Z_\gamma = \frac{\sum_{i=1}^{k} Z_i}{\sqrt{k}}\,. \tag{6}$$

- `method="ssgsea"` (**?**). Single sample GSEA (ssGSEA) calculates a gene set enrichment score per sample as the normalized difference in empirical cumulative distribution functions of gene expression ranks inside and outside the gene set.

By default `method="gsva"` and the `gsva()` function uses the GSVA algorithm.

4 Applications

In this section we illustrate the following applications of GSVA:

- Functional enrichment between two subtypes of leukemia.
- Identification of molecular signatures in distinct glioblastoma subtypes.

Throughout this vignette we will use the C2 collection of curated gene sets that form part of the Molecular Signatures Database (MSigDB) version 3.0. This particular collection of gene sets is provided as a *GeneSetCollection* object called `c2BroadSets` in the accompanying experimental data package GSVAdata, which stores these and other data employed in this vignette. These data can be loaded as follows:

```
> library(GSEABase)
> library(GSVAdata)
> data(c2BroadSets)
> c2BroadSets
```

where we observe that `c2BroadSets` contains 3272 gene sets. We also need to load the following additional libraries:

```
> library(Biobase)
> library(genefilter)
> library(limma)
> library(RColorBrewer)
> library(GSVA)
```

As a final setup step for this vignette, we will employ the `cache()` function from the Biobase package in order to load some pre-computed results and speed up the building time of the vignette:

```
> cacheDir <- system.file("extdata", package="GSVA")
> cachePrefix <- "cache4vignette_"
```

In order to enforce re-calculating everything, either the call to the `cache()` function should be replaced by its first argument, or the following command should be written in the R console at this point:

```
> file.remove(paste(cacheDir, list.files(cacheDir, pattern=cachePrefix), sep="/"))
```

4.1 Functional enrichment

In this section we illustrate how to identify functionally enriched gene sets between two phenotypes. As in most of the applications we start by calculating GSVA enrichment scores and afterwards, we will employ the linear modeling techniques implemented in the limma package to find the enriched gene sets.

The data set we use in this section corresponds to the microarray data from (**?**) which consists of 37 different individuals with human acute leukemia, where 20 of them have conventional childhood acute lymphoblastic leukemia (ALL) and the other 17 are affected with the MLL (mixed-lineage leukemia gene) translocation. This leukemia data set is stored as an `ExpressionSet` object called `leukemia` in the GSVAdata package and details on how the data was pre-processed can be found in the corresponding help page. Enclosed with the RMA expression values we provide some metadata including the main phenotype corresponding to the leukemia sample subtype.

```
> data(leukemia)
> leukemia_eset
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12626 features, 37 samples
  element names: exprs
protocolData
  sampleNames: CL2001011101AA.CEL CL2001011102AA.CEL
```

```
    ... CL2001011152AA.CEL (37 total)
  varLabels: ScanDate
  varMetadata: labelDescription
phenoData
  sampleNames: CL2001011101AA.CEL CL2001011102AA.CEL
    ... CL2001011152AA.CEL (37 total)
  varLabels: subtype
  varMetadata: labelDescription channel
featureData: none
experimentData: use 'experimentData(object)'
Annotation: hgu95a
```

```
> head(pData(leukemia_eset))
```

```
                   subtype
CL2001011101AA.CEL     ALL
CL2001011102AA.CEL     ALL
CL2001011104AA.CEL     ALL
CL2001011105AA.CEL     ALL
CL2001011109AA.CEL     ALL
CL2001011110AA.CEL     ALL
```

```
> table(leukemia_eset$subtype)
```

```
ALL MLL
 20  17
```

Let's examine the variability of the expression profiles across samples by plotting the cumulative distribution of IQR values as shown in Figure **??**. About 50% of the probesets show very limited variability across samples and, therefore, in the following non-specific filtering step we remove this fraction from further analysis.

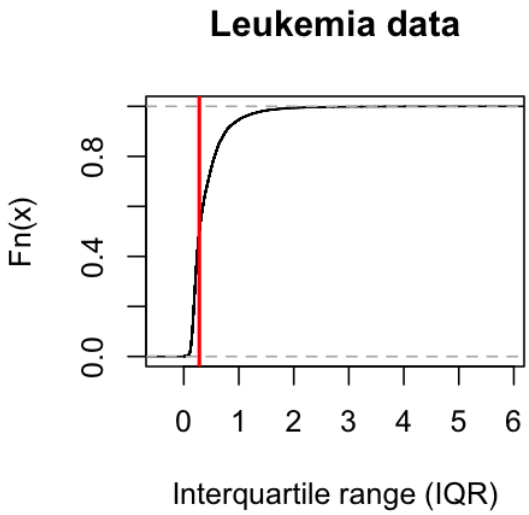


Figure 2: Empirical cumulative distribution of the interquartile range (IQR) of expression values in the leukemia data. The vertical red bar is located at the 50% quantile value of the cumulative distribution.

We carry out a non-specific filtering step by discarding the 50% of the probesets with smaller variability, probesets without Entrez ID annotation, probesets whose associated Entrez ID is duplicated in the annotation, and Affymetrix quality control probes:

```
> filtered_eset <- nsFilter(leukemia_eset, require.entrez=TRUE, remove.dupEntrez=TRUE,
+                           var.func=IQR, var.filter=TRUE, var.cutoff=0.5, filterByQuantile=TRUE,
+                           feature.exclude="^AFFX")
> filtered_eset

$eset
ExpressionSet (storageMode: lockedEnvironment)
assayData: 4297 features, 37 samples
  element names: exprs
protocolData
  sampleNames: CL2001011101AA.CEL CL2001011102AA.CEL
    ... CL2001011152AA.CEL (37 total)
  varLabels: ScanDate
  varMetadata: labelDescription
phenoData
  sampleNames: CL2001011101AA.CEL CL2001011102AA.CEL
    ... CL2001011152AA.CEL (37 total)
  varLabels: subtype
  varMetadata: labelDescription channel
featureData: none
experimentData: use 'experimentData(object)'
Annotation: hgu95a

$filter.log
$filter.log$numDupsRemoved
[1] 2859

$filter.log$numLowVar
[1] 4298

$filter.log$numRemoved.ENTREZID
[1] 1153

$filter.log$feature.exclude
[1] 19

> leukemia_filtered_eset <- filtered_eset$eset
```

The calculation of GSVA enrichment scores is performed in one single call to the `gsva()` function. However, one should take into account that this function performs further non-specific filtering steps prior to the actual calculations. On the one hand, it matches gene identifiers between gene sets and gene expression values. On the other hand, it discards gene sets that do not meet minimum and maximum gene set size requirements specified with the arguments `min.sz` and `max.sz`, respectively, which, in the call below, are set to 10 and 500 genes. Because we want to use limma on the resulting GSVA enrichment scores, we leave deliberately unchanged the default argument `mx.diff=TRUE` to obtain approximately normally distributed ES.

```
> cache(leukemia_es <- gsva(leukemia_filtered_eset, c2BroadSets,
+                           min.sz=10, max.sz=500, verbose=TRUE)$es.obs,
+                           dir=cacheDir, prefix=cachePrefix)
```

We test whether there is a difference between the GSVA enrichment scores from each pair of phenotypes using a simple linear model and moderated t-statistics computed by the `limma` package using an empirical Bayes shrinkage method (see **?**). We are going to examine both, changes at gene level and changes at pathway level and since, as we shall see below, there are plenty of them, we are going to employ the following stringent cut-offs to attain a high level of statistical and biological significance:

```
> adjPvalueCutoff <- 0.001
> logFCcutoff <- log2(2)
```

where we will use the latter only for the gene-level differential expression analysis.

```
> design <- model.matrix(~ factor(leukemia_es$subtype))
> colnames(design) <- c("ALL", "MLLvsALL")
> fit <- lmFit(leukemia_es, design)
> fit <- eBayes(fit)
> allGeneSets <- topTable(fit, coef="MLLvsALL", number=Inf)
> DEgeneSets <- topTable(fit, coef="MLLvsALL", number=Inf,
+                        p.value=adjPvalueCutoff, adjust="BH")
> res <- decideTests(fit, p.value=adjPvalueCutoff)
> summary(res)
```

```
    ALL MLLvsALL
-1    3        9
0  2030     1997
1     1       28
```

Thus, there are 37 MSigDB C2 curated pathways that are differentially activated between MLL and ALL at 0.1% FDR. When we carry out the corresponding differential expression analysis at gene level:

```
> logFCcutoff <- log2(2)
> design <- model.matrix(~ factor(leukemia_eset$subtype))
> colnames(design) <- c("ALL", "MLLvsALL")
> fit <- lmFit(leukemia_filtered_eset, design)
> fit <- eBayes(fit)
> allGenes <- topTable(fit, coef="MLLvsALL", number=Inf)
> DEgenes <- topTable(fit, coef="MLLvsALL", number=Inf,
+                     p.value=adjPvalueCutoff, adjust="BH", lfc=logFCcutoff)
> res <- decideTests(fit, p.value=adjPvalueCutoff, lfc=logFCcutoff)
> summary(res)
```

```
    ALL MLLvsALL
-1    0       71
0     0     4175
1  4297       51
```

Here, 122 genes show up as being differentially expressed with a minimum fold-change of 2 at 0.1% FDR. We illustrate the genes and pathways that are changing by means of volcano plots (Fig. **??**.

The signatures of both, the differentially activated pathways reported by the GSVA analysis and of the differentially expressed genes are shown in Figures **??** and **??**, respectively. Many of the gene sets and pathways reported in Figure **??** are directly related to ALL and MLL.

4.2 Molecular signature identification

In (**?**) four subtypes of Glioblastoma multiforme (GBM) - proneural, classical, neural and mesenchymal - were identified by the characterization of distinct gene-level expression patterns. Using eight gene set signatures specific to brain cell types - astrocytes, oligodendrocytes, neurons and cultured astroglial cells - derived from murine models by (**?**), we replicate the analysis of (**?**) by employing GSVA to transform the gene expression measurements into enrichment scores for these eight gene sets, without taking the sample subtype grouping into account. We start by loading and have a quick glance to the data which forms part of the `GSVAdata` package:

```
> data(gbm_VerhaakEtAl)
> gbm_eset
```

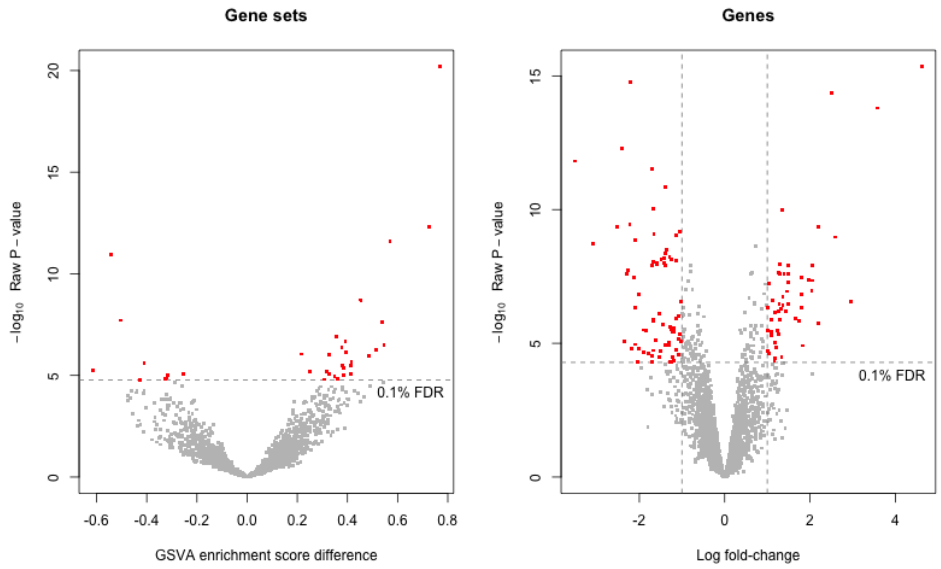


Figure 3: Volcano plots for differential pathway activation (left) and differential gene expression (right) in the leukemia data set.

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 11861 features, 173 samples
  element names: exprs
protocolData: none
phenoData
  rowNames: TCGA.02.0003.01A.01 TCGA.02.0010.01A.01
    ... TCGA.12.0620.01A.01 (173 total)
  varLabels: subtype
  varMetadata: labelDescription channel
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

```
> head(featureNames(gbm_eset))
```

```
[1] "AACS"    "FSTL1"   "ELMO2"   "CREB3L1" "RPS11"
[6] "PNMA1"
```

```
> table(gbm_eset$subtype)
```

```
  Classical Mesenchymal      Neural   Proneural
         38          56          26          53
```

```
> data(brainTxDbSets)
> sapply(brainTxDbSets, length)
```

```
      astrocytic_up       astrocytic_dn        astroglia_up
                 85                  15                  88
       astroglia_dn         neuronal_up         neuronal_dn
                 12                  98                  30
oligodendrocytic_up oligodendrocytic_dn
                 70                  30
```

```
> lapply(brainTxDbSets, head)
```

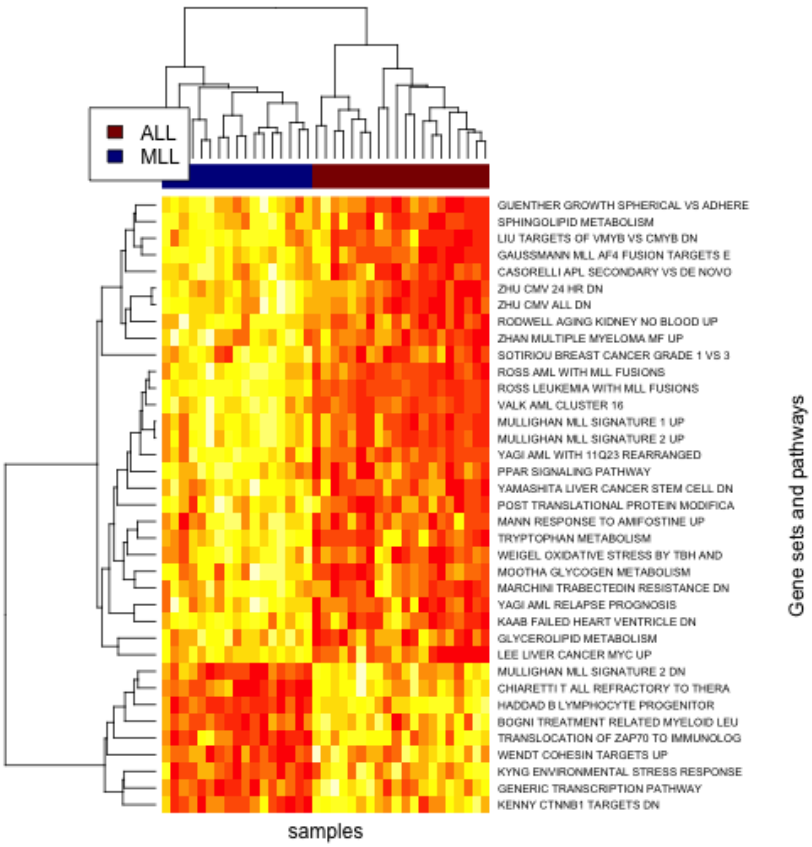


Figure 4: Heatmap of differentially activated pathways at 0.1% FDR in the Leukemia data set.

```
$astrocytic_up
[1] "GRHL1"   "GPAM"    "PAPSS2"  "MERTK"   "BTG1"
[6] "SLC46A1"

$astrocytic_dn
[1] "NPAL3"  "ATP1A1" "FRMD5"  "ASNS"   "SEMA3E" "LPGAT1"

$astroglia_up
[1] "BST2"     "SERPING1" "ACTA2"    "C9orf167" "C1orf31"
[6] "ANXA4"

$astroglia_dn
[1] "PCDH8"   "ATP8A1"  "PHACTR3" "PCDH17"  "CCDC28B"
[6] "TDG"

$neuronal_up
[1] "STXBP1"  "JPH4"    "CACNG3"  "BRUNOL6" "CLSTN2"
[6] "FAM123C"

$neuronal_dn
[1] "DKK3"    "LPHN2"   "AHR"     "NRP1"    "MAP3K15"
[6] "GALNTL4"

$oligodendrocytic_up
[1] "DCT"    "ZNF536" "GNG8"   "ELOVL6" "NR2C1"  "RCBTB1"
```

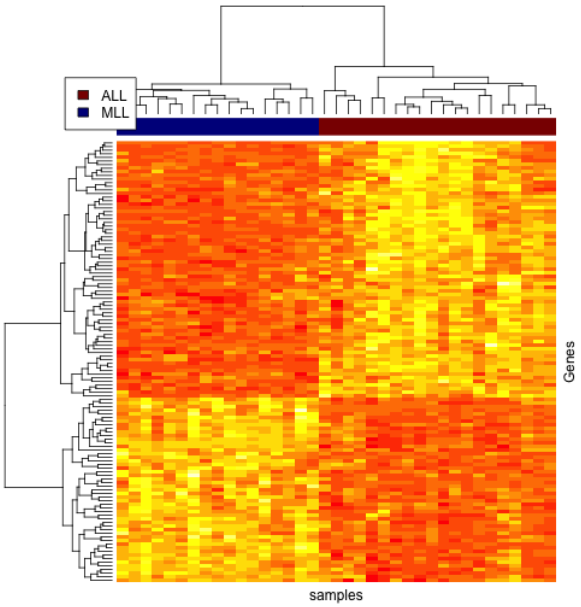


Figure 5: Heatmap of differentially expressed genes with a minimum fold-change of 2 at 0.1% FDR in the leukemia data set.

```
$oligodendrocytic_dn
[1] "DKK3"    "LPHN2"   "AHR"     "NRP1"    "MAP3K15"
[6] "GALNTL4"
```

GSVA enrichment scores for the gene sets contained in `brainTxDbSets` are calculated, in this case using `mx.diff=FALSE`, as follows:

```
> gbm_es <- gsva(gbm_eset, brainTxDbSets, mx.diff=FALSE, verbose=FALSE, parallel.sz=1)$es.obs
```

Figure **??** shows the GSVA enrichment scores obtained for the up-regulated gene sets across the samples of the four GBM subtypes. As expected, the *neural* class is associated with the neural gene set and the astrocytic gene sets. The *mesenchymal* subtype is characterized by the expression of mesenchymal and microglial markers, thus we expect it to correlate with the astroglial gene set. The *proneural* subtype shows high expression of oligodendrocytic development genes, thus it is not surprising that the oligodendrocytic gene set is highly enriched for ths group. Interestingly, the *classical* group correlates highly with the astrocytic gene set. In summary, the resulting GSVA enrichment scores recapitulate accurately the molecular signatures from **?**.

5 Comparison with other methods

In this section we compare with simulated data the performance of GSVA with other methods producing pathway summaries of gene expression, concretely, PLAGE, the combined z-score and ssGSEA which are available through the argument `method` of the function `gsva()`. We employ the following simple linear additive model for simulating normalized microarray data on p genes and n samples divided in two groups representing a case-control scenario:

$$y_{ij} = \alpha_i + \beta_j + \epsilon_{ij}\,, \tag{7}$$

where $\alpha_i \sim \mathcal{N}(0,1)$ is a gene-specific effect, such as a probe-effect, with $i = 1,\ldots,p$, $\beta_j \sim \mathcal{N}(\mu_j, 0.5)$ is a sample-effect with $j = 1, 2$ and $e_{ij} \sim \mathcal{N}(0,1)$ corresponds to random noise.

We will assess the statistical power to detect one differentially expressed (DE) gene set formed by 30 genes, out of $p = 1000$, as a function of the sample size and two varying conditions: the fraction of

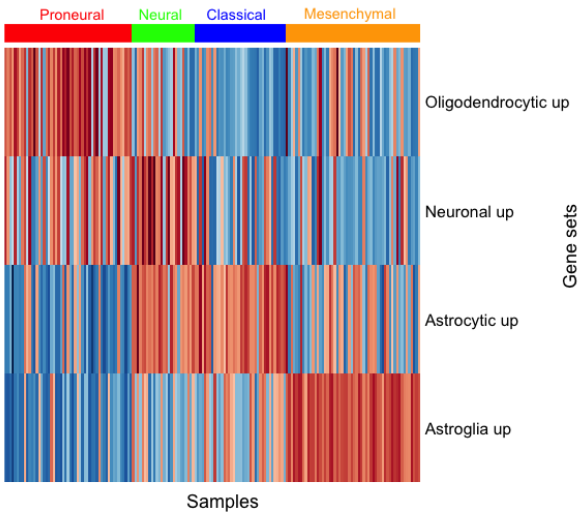


Figure 6: Heatmap of GSVA scores for cell-type brain signatures from murine models (y-axis) across GBM samples grouped by GBM subtype.

differentially expressed genes in the gene set (50% and 80%) and the signal-to-noise ratio expressed as the magnitude of the mean sample effect for one of the sample groups ($\mu_1 = 0$ and either $\mu_2 = 0.5$ or $\mu_2 = 1$). Simulatenously, we will assess the empirical type-I error rate by building using one gene set of 30 non-DE genes.

The following function enables simulating such data, computes the corresponding GSE scores with each method, performs a t-test on the tested gene sets between the two groups of samples for each method and returns the corresponding p-values:

```
> runSim <- function(p, n, gs.sz, S2N, fracDEgs) {
+   sizeDEgs <- round(fracDEgs * gs.sz)
+   group.n <- round(n / 2)
+
+   sampleEffect <- rnorm(n, mean=0, sd=1)
+   sampleEffectDE <- rnorm(n, mean=S2N, sd=0.5)
+   probeEffect <- rnorm(p, mean=0, sd=1)
+   noise <- matrix(rnorm(p*n, mean=0, sd=1), nrow=p, ncol=n)
+   noiseDE <- matrix(rnorm(p*n, mean=0, sd=1), nrow=p, ncol=n)
+   M <- outer(probeEffect, sampleEffect, "+") + noise
+   M2 <- outer(probeEffect, sampleEffectDE, "+") + noiseDE
+   M[1:sizeDEgs, 1:group.n] <- M2[1:sizeDEgs, 1:group.n]
+
+   rownames(M) <- paste0("g", 1:nrow(M))
+   geneSets <- list(H1GeneSet=paste0("g", 1:(gs.sz)),
+                    H0GeneSet=paste0("g", (gs.sz+1):(2*gs.sz)))
+
+   es.gsva <- gsva(M, geneSets, verbose=FALSE, parallel.sz=1)$es.obs
+   es.ss <- gsva(M, geneSets, method="ssgsea", verbose=FALSE, parallel.sz=1)
+   es.z <- gsva(M, geneSets, method="zscore", verbose=FALSE, parallel.sz=1)
+   es.plage <- gsva(M, geneSets, method="plage", verbose=FALSE, parallel.sz=1)
+
+   h1.gsva.pval <- t.test(es.gsva["H1GeneSet", 1:group.n],es.gsva["H1GeneSet", (group.n+1):n])$p.va
+   h1.ssgsea.pval <- t.test(es.ss["H1GeneSet", 1:group.n],es.ss["H1GeneSet", (group.n+1):n])$p.valu
+   h1.zscore.pval <- t.test(es.z["H1GeneSet", 1:group.n],es.z["H1GeneSet", (group.n+1):n])$p.value
+   h1.plage.pval <- t.test(es.plage["H1GeneSet", 1:group.n],es.plage["H1GeneSet", (group.n+1):n])$p
+
```

```
+   h0.gsva.pval <- t.test(es.gsva["H0GeneSet", 1:group.n],es.gsva["H0GeneSet", (group.n+1):n])$p.va
+   h0.ssgsea.pval <- t.test(es.ss["H0GeneSet", 1:group.n],es.ss["H0GeneSet", (group.n+1):n])$p.valu
+   h0.zscore.pval <- t.test(es.z["H0GeneSet", 1:group.n],es.z["H0GeneSet", (group.n+1):n])$p.value
+   h0.plage.pval <- t.test(es.plage["H0GeneSet", 1:group.n],es.plage["H0GeneSet", (group.n+1):n])$p
+
+   c(h1.gsva.pval, h1.ssgsea.pval, h1.zscore.pval, h1.plage.pval,
+     h0.gsva.pval, h0.ssgsea.pval, h0.zscore.pval, h0.plage.pval)
+ }
```

The next function takes the p-values of the output of the previous function and estimates the statistical power as the fraction of non-rejections on the DE gene set, and the empirical type-I error rate as the fraction of rejections on the non-DE gene set, at a significant level $\alpha = 0.05$.

```
> estPwrTypIerr <- function(pvals, alpha=0.05) {
+   N <- ncol(pvals)
+   c(1 - sum(pvals[1, ] > alpha)/N, 1 - sum(pvals[2, ] > alpha)/N,1 - sum(pvals[3, ] > alpha)/N, 1
+         sum(pvals[5, ] <= alpha)/N, sum(pvals[6, ] <= alpha)/N, sum(pvals[7, ] <= alpha)/N, sum(pv
+ }
```

Finally, we perform the simulation on each of the four described scenarios 60 times using the code below. The results in Fig. **??** show that GSVA attains higher statistical power than the other three methods in each of the simulated scenarios, while providing a similar control of the type-I error rate. Note that the fluctuations of this latter estimate are due to the limited number of times we repeat the simulation; see **?** for more stable estimates obtained with a much larger number of repeated simulations.

```
> set.seed(1234)
> exp1 <- cbind(estPwrTypIerr(replicate(60, runSim(1000, 10, gs.sz=30, S2N=0.5, fracDEgs=0.5))),
+               estPwrTypIerr(replicate(60, runSim(1000, 20, gs.sz=30, S2N=0.5, fracDEgs=0.5))),
+               estPwrTypIerr(replicate(60, runSim(1000, 40, gs.sz=30, S2N=0.5, fracDEgs=0.5))),
+               estPwrTypIerr(replicate(60, runSim(1000, 60, gs.sz=30, S2N=0.5, fracDEgs=0.5))))
> exp2 <- cbind(estPwrTypIerr(replicate(60, runSim(1000, 10, gs.sz=30, S2N=1.0, fracDEgs=0.5))),
+               estPwrTypIerr(replicate(60, runSim(1000, 20, gs.sz=30, S2N=1.0, fracDEgs=0.5))),
+               estPwrTypIerr(replicate(60, runSim(1000, 40, gs.sz=30, S2N=1.0, fracDEgs=0.5))),
+               estPwrTypIerr(replicate(60, runSim(1000, 60, gs.sz=30, S2N=1.0, fracDEgs=0.5))))
> exp3 <- cbind(estPwrTypIerr(replicate(60, runSim(1000, 10, gs.sz=30, S2N=0.5, fracDEgs=0.8))),
+               estPwrTypIerr(replicate(60, runSim(1000, 20, gs.sz=30, S2N=0.5, fracDEgs=0.8))),
+               estPwrTypIerr(replicate(60, runSim(1000, 40, gs.sz=30, S2N=0.5, fracDEgs=0.8))),
+               estPwrTypIerr(replicate(60, runSim(1000, 60, gs.sz=30, S2N=0.5, fracDEgs=0.8))))
> exp4 <- cbind(estPwrTypIerr(replicate(60, runSim(1000, 10, gs.sz=30, S2N=1.0, fracDEgs=0.8))),
+               estPwrTypIerr(replicate(60, runSim(1000, 20, gs.sz=30, S2N=1.0, fracDEgs=0.8))),
+               estPwrTypIerr(replicate(60, runSim(1000, 40, gs.sz=30, S2N=1.0, fracDEgs=0.8))),
+               estPwrTypIerr(replicate(60, runSim(1000, 60, gs.sz=30, S2N=1.0, fracDEgs=0.8))))
```

6 GSVA for RNA-Seq data

In this section we illustrate how to use GSVA with RNA-seq data and, more importantly, how the method provides pathway activity profiles analogous to the ones obtained from microarray data by using samples of lymphoblastoid cell lines (LCL) from HapMap individuals which have been profiled using both technologies **??**. These data form part of the experimental package GSVAdata and the corresponding help pages contain details on how the data were processed. We start loading these data and verifying that they indeed contain expression data for the same genes and samples, as follows:

```
> data(commonPickrellHuang)
> stopifnot(identical(featureNames(huangArrayRMAnoBatchCommon_eset),
+                     featureNames(pickrellCountsArgonneCQNcommon_eset)))
> stopifnot(identical(sampleNames(huangArrayRMAnoBatchCommon_eset),
+                     sampleNames(pickrellCountsArgonneCQNcommon_eset)))
```

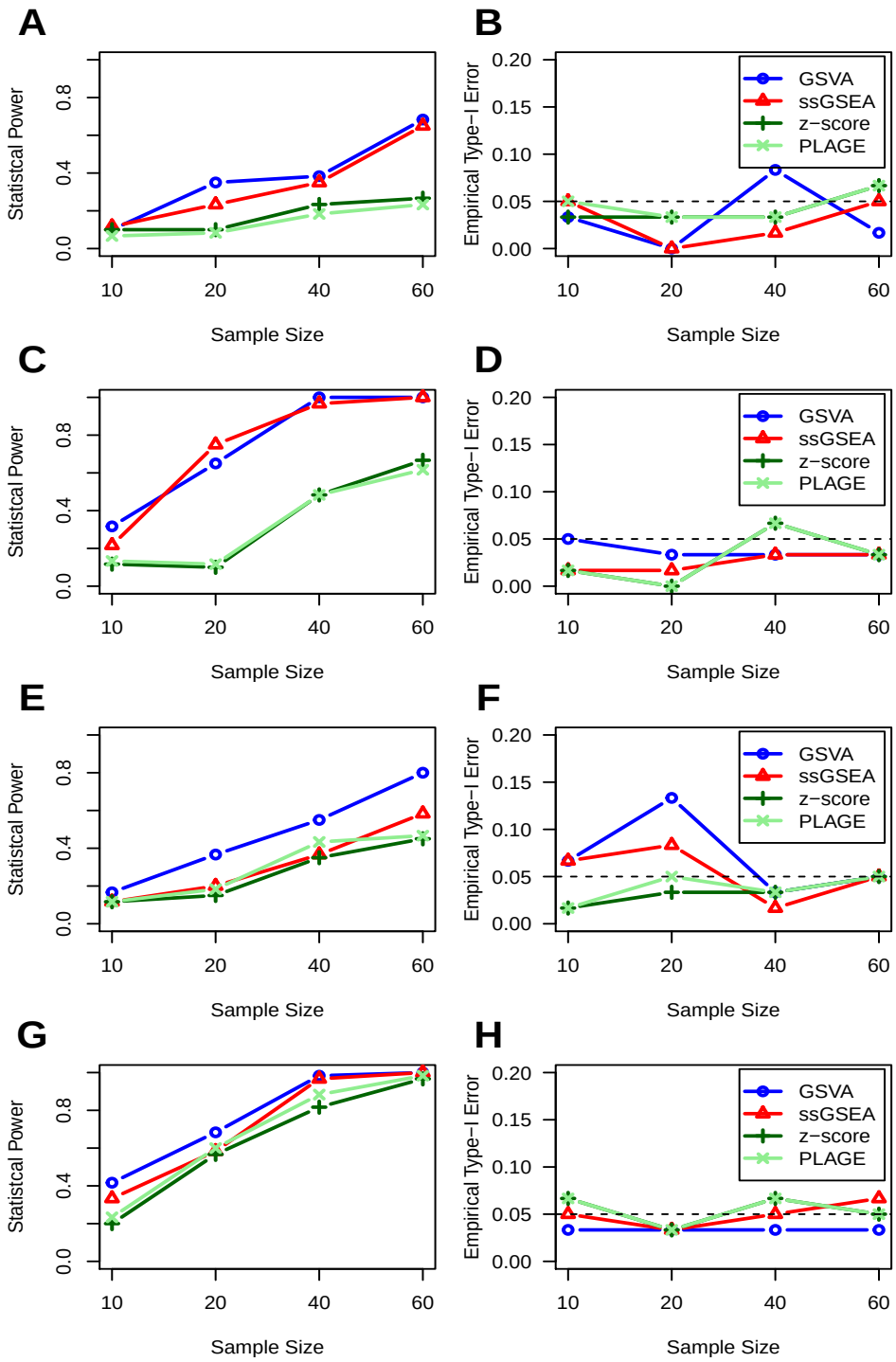


Figure 7: **Comparison of the statistical power and type-I error rate between GSVA, PLAGE, single sample GSEA (ssGSEA) and combined z-score (zscore).** The averaged results of 60 simulations are depicted as function of the sample size on the x-axis, for each of the GSE methods. On the y-axis either the statistical power (A, C, E, G) or the empirical type-I error rate (B, D, F, H) is shown. GSE scores were calculated with each method with respect to two gene sets, one of them differentially expressed (DE) and the other one not. Statistical power and empirical type-I error rates were estimated by performing a t-test on the DE and non-DE gene sets, respectively, at a significance level of $\alpha = 0.05$. These simulations were carried out under the following four different scenarios for the DE gene set: (A,B) weak signal-to-noise ratio, 50% of DE genes in the DE gene set; (C,D) strong signal-to-noise ratio, 50% of DE genes in the DE gene set; (E, F) weak signal-to-noise ratio, 80% of DE genes in the DE gene set; (G, H) strong signal-to-noise ratio, 80% of DE gene in the DE gene set.

Next, for the current analysis we use the subset of canonical pathways from the C2 collection of MSigDB Gene Sets. These correspond to the following pathways from KEGG, REACTOME and BIOCARTA:

```
> canonicalC2BroadSets <- c2BroadSets[c(grep("^KEGG", names(c2BroadSets)),
+                                       grep("^REACTOME", names(c2BroadSets)),
+                                       grep("^BIOCARTA", names(c2BroadSets)))]
> canonicalC2BroadSets
```

```
GeneSetCollection
  names: KEGG_GLYCOLYSIS_GLUCONEOGENESIS, KEGG_CITRATE_CYCLE_TCA_CYCLE, ..., BIOCARTA_ACTINY_PATHWAY
  unique identifiers: 55902, 2645, ..., 8544 (6744 total)
  types in collection:
    geneIdType: EntrezIdentifier (1 total)
    collectionType: BroadCollection (1 total)
```

Additionally, we extend this collection of gene sets with two formed by genes with sex-specific expression:

```
> data(genderGenesEntrez)
> MSY <- GeneSet(msYgenesEntrez, geneIdType=EntrezIdentifier(),
+                collectionType=BroadCollection(category="c2"), setName="MSY")
> MSY
```

```
setName: MSY
geneIds: 266, 84663, ..., 353513 (total: 34)
geneIdType: EntrezId
collectionType: Broad
  bcCategory: c2 (Curated)
  bcSubCategory: NA
details: use 'details(object)'
```

```
> XiE <- GeneSet(XiEgenesEntrez, geneIdType=EntrezIdentifier(),
+                collectionType=BroadCollection(category="c2"), setName="XiE")
> XiE
```

```
setName: XiE
geneIds: 293, 8623, ..., 1121 (total: 66)
geneIdType: EntrezId
collectionType: Broad
  bcCategory: c2 (Curated)
  bcSubCategory: NA
details: use 'details(object)'
```

```
> canonicalC2BroadSets <- GeneSetCollection(c(canonicalC2BroadSets, MSY, XiE))
> canonicalC2BroadSets
```

```
GeneSetCollection
  names: KEGG_GLYCOLYSIS_GLUCONEOGENESIS, KEGG_CITRATE_CYCLE_TCA_CYCLE, ..., XiE (835 total)
  unique identifiers: 55902, 2645, ..., 1121 (6810 total)
  types in collection:
    geneIdType: EntrezIdentifier (1 total)
    collectionType: BroadCollection (1 total)
```

We calculate now GSVA enrichment scores for these gene sets using first the microarray data and then the RNA-seq data. Note that the only requirement to do the latter is to set the argument `rnaseq=TRUE` which is `FALSE` by default.

```
> esmicro <- gsva(huangArrayRMAnoBatchCommon_eset, canonicalC2BroadSets, min.sz=5, max.sz=500,
+                 mx.diff=TRUE, verbose=FALSE, rnaseq=FALSE, parallel.sz=1)$es.obs
> dim(esmicro)
```

```
Features  Samples
     806       36
```

```
> esrnaseq <- gsva(pickrellCountsArgonneCQNcommon_eset, canonicalC2BroadSets, min.sz=5, max.sz=500,
+                  mx.diff=TRUE, verbose=FALSE, rnaseq=TRUE, parallel.sz=1)$es.obs
> dim(esrnaseq)
```

```
Features  Samples
     806       36
```

To compare expression values from both technologies we are going to transform the RNA-seq read counts into RPKM values. For this purpose we need gene length and G+C content information also stored in the GSVAdata package and use the `cpm()` function from the edgeR package. Note that RPKMs can only be calculated for those genes for which the gene length and G+C content information is available:

```
> library(edgeR)
> data(annotEntrez220212)
> head(annotEntrez220212)
```

```
          Length GCcontent
1           2301 0.6292916
10          1344 0.3816964
100         2612 0.5153139
1000        4380 0.4502283
10000       7091 0.3989564
100008586    606 0.4339934
```

```
> cpm <- cpm(exprs(pickrellCountsArgonneCQNcommon_eset))
> dim(cpm)
```

```
[1] 11508    36
```

```
> common <- intersect(rownames(cpm), rownames(annotEntrez220212))
> length(common)
```

```
[1] 11478
```

```
> rpkm <- sweep(cpm[common, ], 1, annotEntrez220212[common, "Length"] / 10^3, FUN="/")
> dim(rpkm)
```

```
[1] 11478    36
```

```
> dim(huangArrayRMAnoBatchCommon_eset[rownames(rpkm), ])
```

```
Features  Samples
   11478       36
```

We finally calculate Spearman correlations between gene and gene-level expression values and gene set level GSVA enrichment scores produced from data obtained by microarray and RNA-seq technologies:

```
> corsrowsgene <- sapply(1:nrow(huangArrayRMAnoBatchCommon_eset[rownames(rpkm), ]),
+                        function(i, expmicro, exprnaseq) cor(expmicro[i, ], exprnaseq[i, ], method=
+                        exprs(huangArrayRMAnoBatchCommon_eset[rownames(rpkm), ]), log2(rpkm+0.1))
> names(corsrowsgene) <- rownames(rpkm)
> corsrowsgs <- sapply(1:nrow(esmicro),
+                      function(i, esmicro, esrnaseq) cor(esmicro[i, ], esrnaseq[i, ], method="spear
+                      exprs(esmicro), exprs(esrnaseq))
> names(corsrowsgs) <- rownames(esmicro)
```

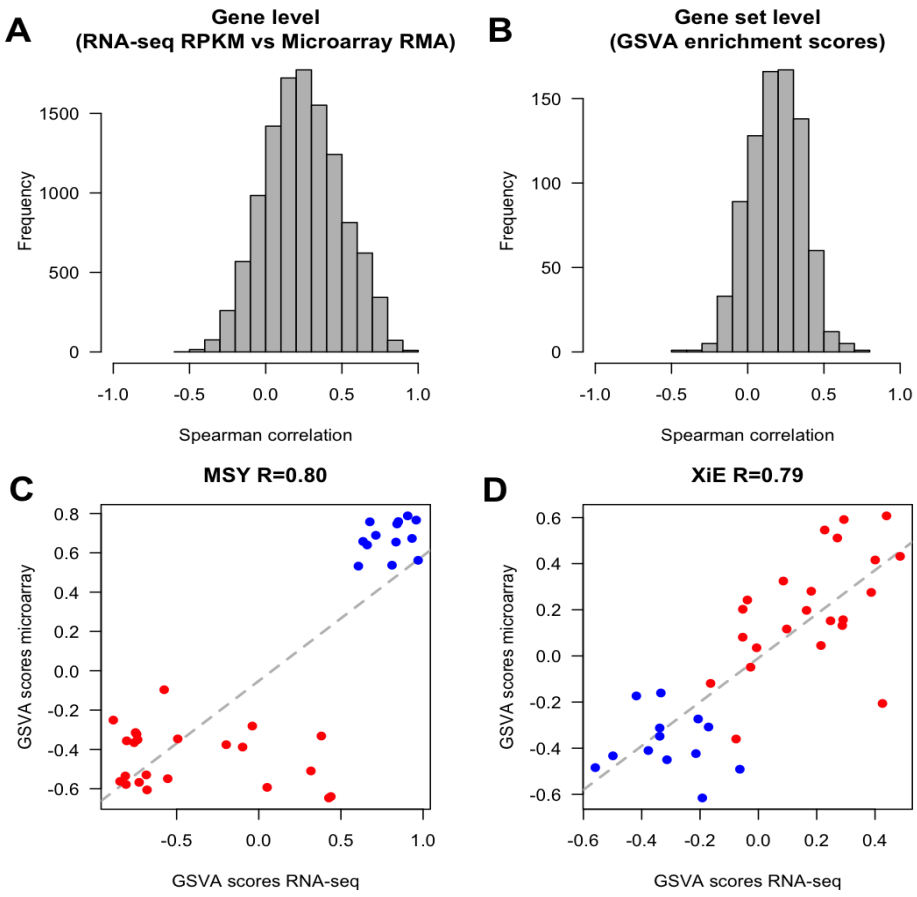


Figure 8: **GSVA for RNA-seq (Argonne).** A. Distribution of Spearman correlation values between gene expression profiles of RNA-seq and microarray data. B. Distribution of Spearman correlation values between GSVA enrichment scores of gene sets calculated from RNA-seq and microarray data. C and D. Comparison of GSVA enrichment scores obtained from microarray and RNA-seq data for two gene sets containing genes with sex-specific expression: MSY formed by genes from the male-specific region of the Y chromosome, thus male-specific, and XiE formed by genes that escape X-inactivation in females, thus female-specific. Red and blue dots represent female and male samples, respectively. In both cases GSVA-scores show very high correlation between the two profiling technologies where female samples show higher enrichment scores in the female-specific gene set and male samples show higher enrichment scores in the male-specific gene set.

In panels A and B of Figure **??** we can see the distribution of these correlations at gene and gene set level. They show that GSVA enrichment scores correlate similarly to gene expression levels produced by both profiling technologies.

We also examined the two gene sets containing gender specific genes in detail: those that escape X-inactivation in female samples (**?**) and those that are located on the male-specific region of the Y chrosomome (**?**). In panels C and D of Figure **??** we can see how microarray and RNA-seq enrichment scores correlate very well in these gene sets, with $\rho = 0.82$ for the male-specific gene set and $\rho = 0.78$ for the female-specific gene set. Male and female samples show higher GSVA enrichment scores in their corresponding gene set. This demonstrates the flexibility of GSVA to enable analogous unsupervised and single sample GSE analyses in data coming from both, microarray and RNA-seq technologies.

7 Session Information

```
> toLatex(sessionInfo())
```

- R version 3.3.2 (2016-10-31), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.36.2, Biobase 2.34.0, BiocGenerics 0.20.0, GSEABase 1.36.0, GSVA 1.22.4, GSVAdata 1.10.0, IRanges 2.8.1, RColorBrewer 1.1-2, S4Vectors 0.12.1, XML 3.98-1.5, annotate 1.52.1, edgeR 3.16.5, genefilter 1.56.0, graph 1.52.0, hgu95a.db 3.2.3, limma 3.30.9, org.Hs.eg.db 3.4.0
- Loaded via a namespace (and not attached): DBI 0.5-1, Matrix 1.2-8, RCurl 1.95-4.8, RSQLite 1.1-2, Rcpp 0.12.9, bitops 1.0-6, digest 0.6.12, grid 3.3.2, lattice 0.20-34, locfit 1.5-9.1, memoise 1.0.0, splines 3.3.2, survival 2.40-1, tools 3.3.2, xtable 1.8-2