GiANT package vignette

Florian Schmid, Christoph Müssel, Johann M. Kraus, Hans A. Kestler

July 21, 2015

Contents

1 Introduction

Gene set analyses investigate the relationship between relevant genes from an experiment and gene sets associated with higher-level terms, such as pathways. For example, sequencing or microarray experiments typically yield a list of genes that are differentially expressed with respect to two experimental conditions, phenotypes or tissues. Gene set analyses can address the question whether these differentially expressed genes are associated to known biological processes or pathways.

Typical gene set analyses are often comprised of a sequence of processing steps [1], for each of which different interchangeable methods exist (see Figure 1). The first of these steps is the calculation of a *gene-level statistic* that quantifies how well each measurement corresponds to the analyzed groups. Examples are the t-statistic, the fold change or the correlation of the measurements to the grouping. In some cases, the gene-level statistic values need to be transformed for the subsequent steps, e.g. by taking the absolute value. The *gene set statistic* summarizes the (transformed) gene-level statistic values for a specific gene set (e.g. a pathway). A computer-intensive test then compares the gene set statistic to a randomly generated baseline distribution to assess whether there is a significant association of the differentially expressed genes to the gene set.

`GiANT` is an R package that implements a dynamic pipeline for gene set analysis based on the described processing steps. It implements various methods for each step as well as predefined pipelines for well-known gene set analysis approaches, but also allows for implementing custom toolchains.

To install `GiANT` in R, type

```
> install.packages("GiANT")
```

To use the package's interfaces to existing methods in other packages, these packages have to be installed as well. For example, `GiANT` comprises wrappers for the `GlobalAncova` method in the `GlobalAncova` package [3] as well as the `gt` method in the `globaltest` package [2]. These suggested packages and their dependencies can be installed using the following commands:

```
> # CRAN packages
> install.packages(c("st", "fdrtool"))
> # Bioconductor packages
> source("http://bioconductor.org/biocLite.R")
> biocLite(c("GlobalAncova", "limma", "DESeq"))
```

After installing all required packages, the `GiANT` package can be loaded via

```
> library(GiANT)
```

2 Applying a gene set analysis

The main interface of the package is the `geneSetAnalysis()` function. In the `analysis` parameter, this function is supplied with a description of the analysis that should be applied. Further arguments hold parameters of the method.

In the following, we apply the well-known Gene Set Enrichment Analysis (GSEA, [6]) to a gene expression data set of 96 breast cancer samples grouped according to their metastasis status and a set of nine pathways that are related to cancer. Both the data set and the pathways are provided in the `GlobalAncova` package. For the `analysis` parameter, we specify the function `analysis.gsea()` that creates a wrapper object describing the toolchain for a GSEA analysis. We further specify that Pearson correlation should be used as a gene-level statistic and that gene set enrichment p-values should be adjusted according to false discovery rate (FDR). We use a significance level of 0.1, which is common when using FDR-based adjustment.

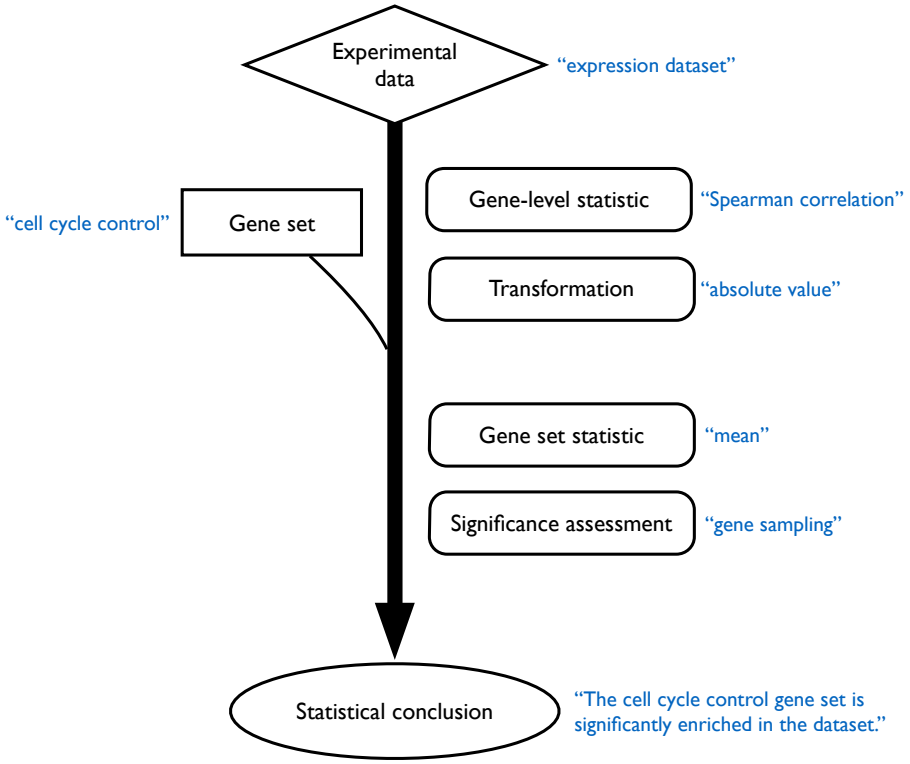


Figure 1: The modular structure of a gene set analysis, implemented in the package. For each module an example is given in red.

```
> # load data
> require(GlobalAncova)
> data(vantVeer)
> data(phenodata)
> data(pathways)
> resGsea <- geneSetAnalysis(
+         labs = phenodata$metastases,
+         method = "pearson",
+         numSamples = 1000,
+         dat = vantVeer,
+         geneSets = pathways,
+         analysis = analysis.gsea(),
+         adjustmentMethod = "fdr",
+         signLevel=0.1)
```

The results of enrichment analyses can be summarized via the `summary()` function:

```
> summary(resGsea)

    Analysis: gsea

    9 gene set(s) tested:
      - 2 gene set(s) with raw p-value < 0.1
      - min p-value: cell_cycle_control (0.005994006)

    Correction for multiple testing: fdr
      - 2 gene set(s) with adjusted p-value < 0.1
```

We can see that only one of the gene sets – the cell cycle control – is significantly enriched. To get more detailed results, we can create a table with the raw and adjusted p-values for each gene set that can be useful for further processing steps:

```
> tab <- createSummaryTable(resGsea)
> tab

                  geneSetName adjustedPValues  rawPValues geneSetSize
1          cell_cycle_control      0.05394605 0.005994006          31
2      notch_delta_signalling      0.07642358 0.016983017          34
3              p53_signalling      0.52147852 0.173826174          33
4               wnt_signaling      0.65409590 0.290709291         176
5    tight_junction_signaling      0.80919081 0.449550450         326
6                  apoptosis      0.87437562 0.777222777         187
7              ras_signalling      0.87437562 0.655344655         266
8          tgf_beta_signaling      0.87437562 0.719280719          82
9 androgen_receptor_signaling      0.95204795 0.952047952          72
```

This function can be parameterized in different ways. E.g., to extract only the significantly enriched gene sets ordered by their name, type

```
> tab <- createSummaryTable(resGsea, significantOnly=TRUE, orderBy="geneSetName")
> tab

             geneSetName adjustedPValues  rawPValues geneSetSize
1     cell_cycle_control      0.05394605 0.005994006          31
2 notch_delta_signalling      0.07642358 0.016983017          34
```

The data frames resulting from `createSummaryTable()` can be used to export results to other programs, e.g. by writing it to CSV files using `write.csv()`.

The results of the enrichment analyses can also be visualized in form of histograms, where each histogram visualizes the gene set enrichment score of one pathway as compared to the null distribution. The following plots the cell cycle control gene set (`subset=3`):

```
> hist(resGsea, subset = 3, aggregate = TRUE)
```

The results are shown in Figure 2. Here, the histogram corresponds to the null distribution of gene set statistic values for random gene sets and class assignments, whereas the red vertical line corresponds to the gene set statistic score of the cell cycle control gene set. The fact that the line is to the right of the 95% quantile of the null distribution (the blue line) expresses that the gene set is significantly enriched.

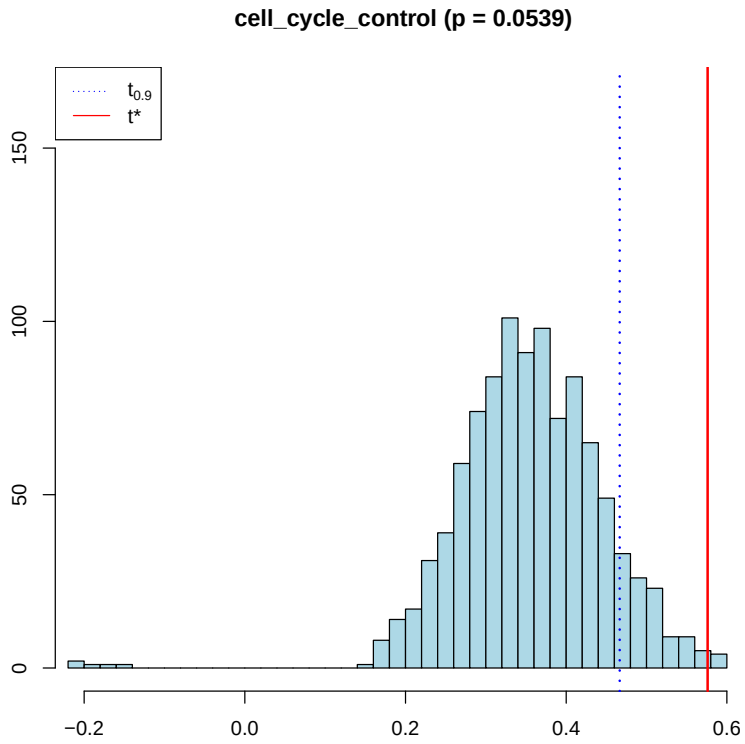


Figure 2: Histogram visualizing the results of a Gene Set Enrichment Analysis for the cell cycle control gene set. The histogram corresponds to the null distribution of the gene set statistic, whereas the red line corresponds to the statistic score of the cell cycle control gene set. The blue line is the 95% quantile of the null distribution.

Another commonly used gene set analysis method is the overrepresentation analysis, which checks whether the genes of a core set are significantly overrepresented among the gene sets. While the GSEA considers all genes in a data set and their corresponding gene-level statistics, an overrepresentation analysis requires a prior selection of an interesting subset, such as those genes whose p-value for differential expression is significant, or the top 100 differentially expressed genes. The names of the genes in this subset are supplied as a vector in the `coreSet` parameter. Here, we select the 25 genes with the highest absolute correlation to the class label:

```
> stat <- abs(apply(vantVeer,1,cor,y = phenodata$metastases))
> coreSet <- rownames(vantVeer)[tail(order(stat), 25)]
> resOverrep <- geneSetAnalysis(
+         dat = vantVeer,
+         geneSets = pathways[1:4],
+         analysis = analysis.customOverrepresentation(),
+         coreSet = coreSet,
+         adjustmentMethod = "fdr",
+         signLevel=0.1)
> summary(resOverrep)

    Analysis: overrepresentation

    4 gene set(s) tested:
      - 1 gene set(s) with raw p-value < 0.1
      - min p-value: cell_cycle_control (3.4487e-05)

    Correction for multiple testing: fdr
      - 1 gene set(s) with adjusted p-value < 0.1
```

We can see that the cell cycle control gene set, which was already significantly enriched in the GSEA, is significantly overrepresented in the core set as well.

For the overrepresentation analysis, `plotOverrepresentation()` can be used to visualize the overlaps of the core set and the specified gene sets in form of a Venn diagram:

```
> plotOverrepresentation(resOverrep, aggregate = TRUE)
```

The plot is shown in Figure 3. Venn diagrams can only be plotted for up to five sets, which is why we restrict the overrepresentation analysis to four sets here. For more than 5 sets or area proportional Venn diagramms we recommend VennMaster [4, 5] (http://sysbio.uni-ulm.de/?Software:VennMaster). The cell cycle control gene set, which is the only significant set, has an overlap of 6 genes with the core set. The other gene sets show almost no overlap with the core set and with each other.

Apart from GSEA and overrepresentation analysis, the package defines further standard analyses, such as a wrapper for the global Ancova method in the `GlobalAncova` package. All these methods can be used in the same way. An overview of predefined configurations can be found on the help page `predefinedAnalyses`.

3 Gene set uncertainty analysis

In some cases, gene sets may not stem from validated sources like manually curated ontologies, but may be hand-crafted for the specific question under consideration. In this scenario, it is not only interesting whether the gene set is significantly enriched among the measurements: Another important aspect is the quality of the gene set itself. One way of quantifying gene set quality is a robustness test that performs multiple gene set analyses with slightly perturbed gene sets and checks whether the gene set statistic of the original gene set differs significantly from those of the

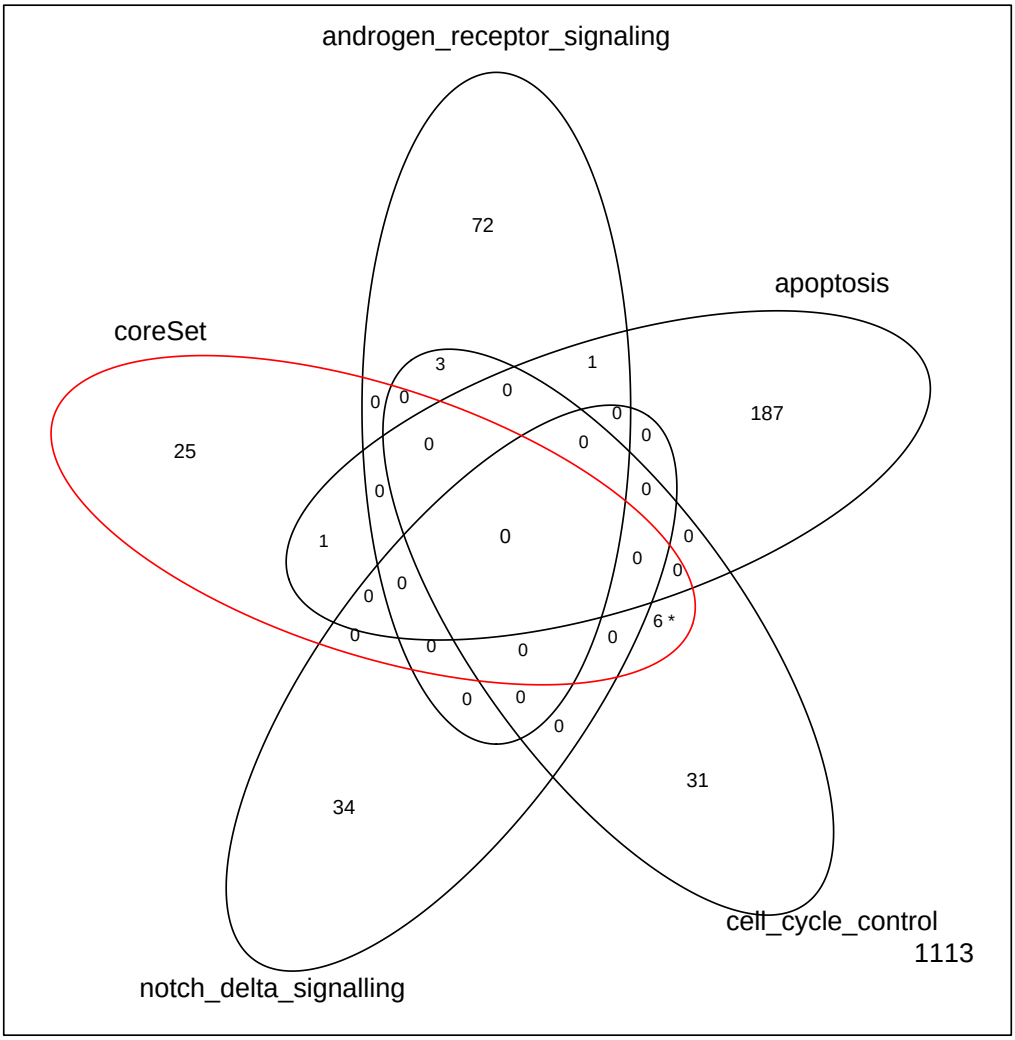


Figure 3: Venn diagram showing the overlaps of gene sets with the analyzed core set of differentially expressed genes. The numbers correspond to the numbers of genes in the sets and in the overlaps respectively.

perturbed sets. This assesses how susceptible the gene set is to exchanging members and thus measures the uncertainty (or certainty) of the gene set. `GiANT` implements this strategy in the `evaluateGeneSetUncertainty()` function.

For example, the following quantifies the uncertainty of the cell cycle control gene set that was significantly enriched in the above analyses:

```
> resUncertainty <- evaluateGeneSetUncertainty(
+         #parameters in ...
+         labs = phenodata$metastases,
+         numSamples = 1000,
+         #parameters for evaluateGeneSetUncertainty
+         dat = vantVeer,
+         geneSet = pathways[[3]],
+         analysis = analysis.averageCorrelation(),
+         numSamplesUncertainty = 100,
+         k = seq(0.1,0.9,by = 0.1))
> plot(resUncertainty,
+         main = names(pathways[3]),
+         addMinimalStability = TRUE)
```

Figure 3 shows the results of the analysis. The blue lines show the tested degrees of fuzziness. The dots in each column give the quantiles ($\{0.05, 0.5, 0.95\}$) of the test statistic values obtained by resampling a percentage of genes from the gene set (k) and the remaining genes ($1 - k$) from the set of all genes in the dataset. The values for $k = 0$ give the quantiles of the null distribution with the green line corresponding to the 95% quantile. The red line shows the value of the test-statistic for the original set. The fuzziness of the gene set is the tested value of k which has a non overlapping confidence interval with the null distribution. The dotted lines give the lower bound for the fuzziness of the gene set.

4 Custom analyses

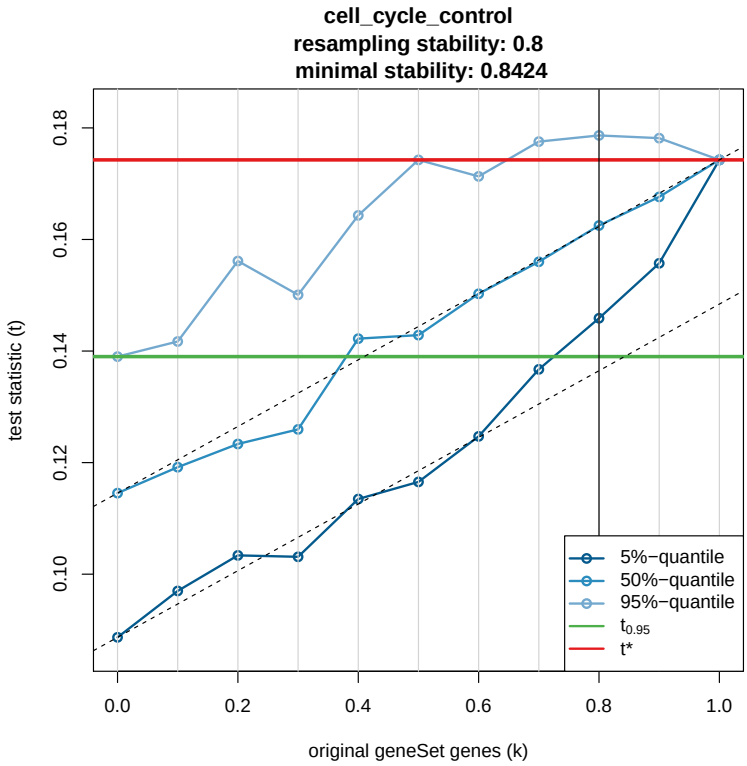
Apart from using predefined analysis configurations, it is also possible to define custom analyses. The `gsAnalysis()` function generates wrapper objects for such analysis pipeline configurations. Generally, methods can either follow the toolchain of gene-level statistic calculation, transformation, gene set statistic calculation and significance assessment, or they can define a single, "global" method performing all required steps at once. In the following, we will define an analysis that rates gene sets according to the average absolute correlation of the involved genes to the class labels. This belongs to the former category, i.e. it can be formulated as a four-step toolchain.

As a first step, we need to define a function that calculates the gene-level statistic, i.e. the correlation to the class labels. Basically, this is a wrapper that applies R's `cor()` function to all rows of a data set:

```
> myGLS <- function(dat, labs, method = "pearson"){
+         return(apply(dat, 1, function(x){
+                         cor(x = x, y = labs, method = method)
+                 }))
+ }
```

In the next step, the correlation values are transformed by taking the absolute value. We do not need to define a custom function here, as this is exactly what R's `abs()` function does.

The gene set statistic is simply the average of the absolute correlation values for all genes in the set:




```
> myGSS <- function(x, geneSetIndices){
+     return(mean(x[geneSetIndices]))
+ }
```

Here, the absolute correlation values for all genes are supplied to `myGSS()` in `x`, and the indices of the genes in the currently investigated gene set are supplied in `geneSetIndices`.

The final step is the significance assessment. This step compares the gene set statistic value of the true gene set to a null distribution that is approximated by random sampling. Instead of writing this function ourselves, we rely on a function that is supplied by the package: `significance.sampling()` draws a high number of random gene sets and calculates the gene set statistic in the same way as for the true gene set. The p-value of the gene set is the fraction of genes having a greater gene set statistic value than the value for the set.

Let us now assemble the above steps. For this purpose, we define a function `myAnalysis()` that calls `gsAnalysis()` with the specified functions and their parameter names internally. Calling `myAnalysis()` function will then return the wrapper object that defines our analysis:

```
> myAnalysis <- function(){
+         return(gsAnalysis(name = "myAnalysis",
+                 gls = "myGLS",
+                 glsParameterNames = c("labs", "method"),
+                 transformation = "abs",
+                 transformationParameterNames = NULL,
+                 gss = "myGSS",
+                 gssParameterNames = NULL,
+                 globalStat = NULL,
+                 globalStatParameterNames = NULL,
+                 significance = "significance.sampling",
```

```
+                 significanceParameterNames = c("numSamples"),
+                 testAlternative = "greater"))
+ }
```

Here, `gls`, `transformation`, `gss` and `significance` define the functions to be called for the gene-level statistic calculation, the transformation, the gene set statistic calculation and the significance assessment respectively. `glsParameterNames`, `transformationParameterNames`, `gssParameterNames` and `significanceParameterNames` define the names of additional parameters of the corresponding functions. These parameters can be supplied in the `...` argument of `geneSetAnalysis()` when performing the analysis. The parameters `globalStat` and `globalStatParameterNames` can be used for the alternative definition of a single, global analysis function discussed above, which is why they are set to `NULL` here. Finally, `testAlternative` specifies how the p-value is determined from the gene set statistic values, i.e. whether the statistic value of a relevant gene set is expected to be greater or less than the random values.

Our new analysis can now be called exactly in the same way as the predefined analysis by supplying it to `geneSetAnalysis()`:

```
> myResult <- geneSetAnalysis(
+         labs = phenodata$metastases,
+         method = "pearson",
+         numSamples = 100,
+         dat = vantVeer,
+         geneSets = pathways,
+         analysis = myAnalysis(),
+         adjustmentMethod = "fdr")
> hist(myResult)
```

Similarly to the predefined significance assessment `significance.sampling()` we used above, there are predefined building blocks for all pipeline steps that can be combined freely. For more information, refer to the help pages `gls`, `transformation`, `gss`, `significance` and `globalAnalysis`.

Finally, it should be mentioned that an analysis comprising the above steps is already defined in the package: Instead of defining our own analysis, we could also have used the predefined analysis `analysis.averageCorrelation()` in this case.

References

[1] Marit Ackermann and Korbinian Strimmer. A general modular framework for gene set enrichment analysis. *BMC Bioinformatics*, 10(1):47, 2009.

[2] Jelle J. Goeman, Sara A. van de Geer, Floor de Kort, and Hans C. van Houwelingen. A global test for groups of genes: testing association with a clinical outcome. *Bioinformatics*, 20(1):93–99, 2004.

[3] Manuela Hummel, Reinhard Meister, and Ulrich Mansmann. Globalancova: exploration and assessment of gene group effects. *Bioinformatics*, 24(1):78–85, 2008.

[4] Hans A. Kestler, André Müller, Thomas M. Gress, and Malte Buchholz. Generalized venn diagrams: a new method of visualizing complex genetic set relations. *Bioinformatics*, 21(8):1592–1595, 2005.

[5] Hans A. Kestler, André Müller, Johann Kraus, Malte Buchholz, Thomas Gress, Hongfang Liu, David Kane, Barry Zeeberg, and John Weinstein. Vennmaster: Area-proportional euler diagrams for functional go analysis of microarrays. *BMC Bioinformatics*, 9(1):67, 2008.

[6] Aravind Subramanian, Pablo Tamayo, Vamsi K. Mootha, Sayan Mukherjee, Benjamin L. Ebert, Michael A. Gillette, Amanda Paulovich, Scott L. Pomeroy, Todd R. Golub, Eric S. Lander, and Jill P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43):15545–15550, 2005.