Comparing Performance of Gene Set Test Methods Using Biologically Relevant Simulated Data

Richard M. Lambert
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

Part of the Applied Statistics Commons

Recommended Citation

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.
COMPARING PERFORMANCE OF GENE SET TEST METHODS USING BIOLOGICALLY RELEVANT SIMULATED DATA

by

Richard M. Lambert

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE in

Statistics

Approved:

John Stevens, Ph.D. Richard Cutler, Ph.D.
Major Professor Committee Member

Yan Sun, Ph.D. Laurens H. Smith, Ph.D.
Committee Member Interim Vice President for Research and Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2018
Abstract

Comparing Performance of Gene Set Test Methods Using Biologically Relevant Simulated Data

by

Richard M. Lambert, Master of Science
Utah State University, 2018

Major Professor: John Stevens, Ph.D.
Department: Mathematics and Statistics

Determining differential gene expression in two or more sample groups is of great biomedical interest in understanding the genetic causes of diseases and health conditions and evaluating efficacy of genetic treatments. Gene set testing is a relatively new method of testing for differential expression between sample groups by creating groups of functionally related genes called gene sets. In this research, we evaluated and compared the statistical power and false discovery rate of the following gene set test methods: mvGST, ROAST, CAMERA, ROMER, GlobalTest, GSA, PAGE, SAFE, sigPathway, and GSEAlm.

We developed a simulation framework to generate datasets that are both biologically relevant and representative of actual gene expression data. We identified several biological parameters of interest and determined realistic values for each of them by either sampling real gene expression data sets or literature review. We then identified 5 interesting parameter pairings and tested each combination of parameter values with either 50 or 100 simulated data sets to determine how power and FDR vary as a function of each parameter as well as identify possible interactions between parameters.
Public Abstract

Comparing Performance of Gene Set Test Methods Using Biologically Relevant Simulated Data

Richard M. Lambert

Today we know that there are many genetically driven diseases and health conditions. These problems often manifest only when a set of genes are either active or inactive. Recent technology allows us to measure the activity level of genes in cells, which we call gene expression. It is of great interest to society to be able to statistically compare the gene expression of a large number of genes between two or more groups. For example, we may want to compare the gene expression of a group of cancer patients with a group of non-cancer patients to better understand the genetic causes of that particular cancer. Understanding these genetic causes could potentially lead to improved treatment options.

Initially, gene expression was tested on a per gene level for statistical difference. In more recent years, it has been determined that grouping genes together by biological processes into gene sets and comparing groups at the gene set level probably makes more sense biologically. A number of gene set test methods have since been developed. It is critically important that we know if these gene set test methods are accurate.

In this research, we compare the accuracy of a group of popular gene set test methods across a range of biologically realistic scenarios. In order to measure accuracy, we need to know whether each gene set is differentially expressed or not. Since this is not possible in real gene expression data, we use simulated data. We develop a simulation framework that generates gene expression data that is representative of actual gene expression data and use it to test each gene set method over a range of biologically relevant scenarios. We then compare the power and false discovery rate of each method across these scenarios.
Acknowledgments

I would like to thank Dr. Stevens for his great support on this project. He allowed me freedom to move forward with my ideas, yet always provided valuable insights and direction at just the right times. I truly could not have succeeded without him.

I would also like to thank and express my gratitude to the Utah Agricultural Experiment Station for their financial support, in association with multi-state project W3112 Reproductive Performance in Domestic Ruminants.

I also acknowledge the Center for High Performance Computing at the University of Utah and thank them for the many, many hours of computation time I used to run so many simulations.

Finally, I would like to thank my family. To my daughters, Alaina and Emma, thank you for entertaining yourselves so well while Dad spent so much time at the computer. To my wife, thank you so much for your constant moral, emotional, and physical support throughout this process.

Richard M. Lambert
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Public Abstract</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Related Work</td>
<td>4</td>
</tr>
<tr>
<td>3 Biological Relevance</td>
<td>9</td>
</tr>
<tr>
<td>3.1 Overview</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Parameters of Interest</td>
<td>9</td>
</tr>
<tr>
<td>3.2.1 Sample Size</td>
<td>10</td>
</tr>
<tr>
<td>3.2.2 Gene Set Size</td>
<td>10</td>
</tr>
<tr>
<td>3.2.3 Gene-wise Expression Variance</td>
<td>13</td>
</tr>
<tr>
<td>3.2.4 Biologically Relevant Differential Expression</td>
<td>13</td>
</tr>
<tr>
<td>3.2.5 Inter-Gene Correlation</td>
<td>14</td>
</tr>
<tr>
<td>3.2.6 Proportion of Differentially Expressed Genes</td>
<td>14</td>
</tr>
<tr>
<td>3.3 Summary</td>
<td>14</td>
</tr>
<tr>
<td>4 Gene Set Test Methods</td>
<td>16</td>
</tr>
<tr>
<td>4.1 Overview</td>
<td>16</td>
</tr>
<tr>
<td>4.2 Hypothesis Type</td>
<td>16</td>
</tr>
<tr>
<td>4.3 P-value Technique</td>
<td>17</td>
</tr>
<tr>
<td>4.4 Summary</td>
<td>18</td>
</tr>
<tr>
<td>5 Simulation Framework</td>
<td>20</td>
</tr>
<tr>
<td>5.1 Overview</td>
<td>20</td>
</tr>
<tr>
<td>5.2 Test Scenarios</td>
<td>20</td>
</tr>
<tr>
<td>5.3 Significance</td>
<td>21</td>
</tr>
<tr>
<td>5.4 Multiple Hypothesis Correction</td>
<td>21</td>
</tr>
<tr>
<td>5.5 Power</td>
<td>22</td>
</tr>
<tr>
<td>5.6 False Discovery Rate</td>
<td>23</td>
</tr>
<tr>
<td>5.7 Framework Components</td>
<td>23</td>
</tr>
<tr>
<td>5.7.1 Initial Setup</td>
<td>24</td>
</tr>
<tr>
<td>5.7.2 Data Generation</td>
<td>24</td>
</tr>
<tr>
<td>5.7.3 Gene Set Tests</td>
<td>24</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Summary Statistics of GEO Experiment Sample Sizes</td>
</tr>
<tr>
<td>3.2</td>
<td>Summary Statistics of Gene Set Sizes from the Human Genome</td>
</tr>
<tr>
<td>3.3</td>
<td>Summary of Parameters of Interest</td>
</tr>
<tr>
<td>4.1</td>
<td>Gene Set Methods</td>
</tr>
<tr>
<td>5.1</td>
<td>Test Scenario Parameter Pairings</td>
</tr>
<tr>
<td>5.2</td>
<td>Multiple Hypothesis Adjustment Methods Used for Each Method</td>
</tr>
<tr>
<td>5.3</td>
<td>List of R Libraries Used</td>
</tr>
<tr>
<td>6.1</td>
<td>Samples to Achieve 90% Power and Control FDR at .05 by Method When Gene Set is 100% DE</td>
</tr>
<tr>
<td>6.2</td>
<td>Samples to Achieve 90% Power and Control FDR at .05 by Method When Gene Set is 25% DE</td>
</tr>
<tr>
<td>6.3</td>
<td>Samples to Achieve 90% Power and Control FDR at .05 by Method for Fold-Change of 1.3</td>
</tr>
<tr>
<td>6.4</td>
<td>Samples to Achieve 90% Power and Control FDR at .05 by Method Across Inter-Gene Correlation</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Distribution of the number of samples per group in 60 randomly selected GEO data sets</td>
<td>11</td>
</tr>
<tr>
<td>3.2</td>
<td>Cumulative density plot of the gene set size from every biological process gene set in the human genome.</td>
<td>12</td>
</tr>
<tr>
<td>5.1</td>
<td>Performance Comparison of Original Version and Optimized Version</td>
<td>29</td>
</tr>
<tr>
<td>6.1</td>
<td>Method Power by Total Number of Genes and Sample Size (1 of 2)</td>
<td>32</td>
</tr>
<tr>
<td>6.2</td>
<td>Method Power by Total Number of Genes and Sample Size (2 of 2)</td>
<td>33</td>
</tr>
<tr>
<td>6.3</td>
<td>Method FDR by Total Number of Genes and Sample Size (1 of 2)</td>
<td>35</td>
</tr>
<tr>
<td>6.4</td>
<td>Method FDR by Total Number of Genes and Sample Size (2 of 2)</td>
<td>36</td>
</tr>
<tr>
<td>6.5</td>
<td>Method Type Power by Total Number of Genes and Sample Size</td>
<td>38</td>
</tr>
<tr>
<td>6.6</td>
<td>Method Type FDR by Total Number of Genes and Sample Size</td>
<td>39</td>
</tr>
<tr>
<td>6.7</td>
<td>Method Power by Proportion Differentially Expressed and Sample Size (1 of 2)</td>
<td>41</td>
</tr>
<tr>
<td>6.8</td>
<td>Method Power by Proportion Differentially Expressed and Sample Size (2 of 2)</td>
<td>42</td>
</tr>
<tr>
<td>6.9</td>
<td>Method FDR by Proportion Differentially Expressed and Sample Size (1 of 2)</td>
<td>44</td>
</tr>
<tr>
<td>6.10</td>
<td>Method FDR by Proportion Differentially Expressed and Sample Size (2 of 2)</td>
<td>45</td>
</tr>
<tr>
<td>6.11</td>
<td>Method Type Power by Sample Size and Proportion DE</td>
<td>47</td>
</tr>
<tr>
<td>6.12</td>
<td>Method Type FDR by Sample Size and Proportion DE</td>
<td>48</td>
</tr>
<tr>
<td>6.13</td>
<td>Method Power by Fold-Change and Sample Size (1 of 2)</td>
<td>49</td>
</tr>
<tr>
<td>6.14</td>
<td>Method Power by Fold-Change and Sample Size (2 of 2)</td>
<td>50</td>
</tr>
<tr>
<td>6.15</td>
<td>Method FDR by Fold-Change and Sample Size (1 of 2)</td>
<td>51</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In simple terms, the central dogma of molecular biology states that DNA in cells gets transcribed into RNA which gets translated into proteins [1, 2] which regulate or govern biological functions. An individual gene is a subsection of the entire DNA sequence that represents a certain biological function. For example, one gene may be involved in cell differentiation and another in apoptosis. At different stages of a cell’s development and life cycle, the activity level of each gene, or gene expression, is controlled by a process called gene regulation.

Microarrays can measure the gene expression levels of a large number of genes simultaneously. By comparing gene expression levels of normal subjects with those of subjects with some disease or condition, we can try to understand genetic causes of diseases and health conditions. Furthermore, potential treatments can similarly be tested at the gene expression level to see if they are addressing the underlying genetic causes. Therefore, there is great value in interpreting gene expression data.

Gene expression datasets often have thousands or tens of thousands of variables (p genes) and only a few observations (n subjects). Traditional statistical methods often fail on high-dimensional data (when \( p \gg n \)). To resolve this issue, a new type of method called gene set testing was developed. The gene set test idea more or less originated with the Gene Set Enrichment Analysis method, commonly referred to as GSEA [3]. Since then, many other gene set test methods have been developed.

The basic idea of gene set testing is that genes with similar function are grouped together into clusters called gene sets and a statistical test is performed at the gene set level instead of the gene level. This greatly reduces the dimensionality of the data for the statistical test being used. In practice, gene sets are usually created based on biological
processes, as annotated in a public gene database such as the Gene Ontology (GO) [4, 5] or KEGG [6]. In a sense, each gene set is a proxy for a particular biological process.

Once the gene sets have been created, a statistical test is performed at the gene set level instead of the individual gene level. For any particular gene set, if there is a statistically significant difference in gene expression levels between any two test groups, we say that there is differential expression (DE) between those groups for that gene set. A significant result also implies that the activity level of the biological process represented by the gene set is different between subject groups.

In addition to GSEA, many other gene set methods have been developed such as GSEAlm (GSEA linear model) [7], ROAST [8], CAMERA [9], ROMER [10] GlobalTest [11], mvGST [12], PANTHER [13], sigPathway [14], SAFE [15], PAGE [16], GSA [17], SAM-GS [18], and Ingenuity Pathway Analysis [19]. The goal of each of these gene set test methods is the same, to detect differential expression of gene sets, but the statistical methodologies of each method are quite different. Therefore, it is of interest to see which methods perform the best in detecting true positives while avoiding false positives for various biologically realistic and relevant scenarios.

The purpose of this research was to investigate and explore the statistical performance of a number of gene set test methods by simulating gene expression data where we controlled which gene sets were differentially expressed (DE) and which were just noise. A number of test scenarios were devised that represented realistic gene expression data and experiment designs. For each scenario, we generated 100 data sets and tested each of them with each gene set method. With each method, we recorded the power (rate at which the generated DE gene sets were actually found to be significant) and the false discovery rate (FDR, proportion of gene sets found to be significant that were actually non-DE).

In chapter 2, we review previous work that involved comparing gene set test methods. In particular, we will focus on those that used simulated data to compare methods. With the exception of Fridley et al. [20], there has been little comparable work in doing large scale simulation studies across gene set methods.
Another aspect that is lacking in the current gene set simulation literature is whether the test scenarios are biologically meaningful or whether the generated data is biologically realistic. If the test scenarios are unrealistic, the theoretical power of a method is of little value to the scientists who might actually use the method. In chapter 3, we establish biological relevance of the test scenarios we designed and the realism of the data we generated.

Chapter 4 introduces and briefly describes the 18 gene set test methods (10 core methods and 8 variations) we tested in this research. It includes the type and technique of each method which will be used in the final analysis.

To perform this research, a test framework was developed in the R programming language [21] to simulate the datasets of gene expression data, run each data set through each of the gene set test methods, record the power and FDR of each method, and generate interesting plots. Chapter 5 describes this R framework in great detail. It also provides the details of the specific scenarios that were tested.

The simulation results for each of the test scenarios are then presented and discussed in detail in chapter 6. Each section covers a different pairing of biological parameters and includes a description of the scenario, lattice plots of power and FDR, and an analysis of the results.

Finally, chapter 7 concludes by summarizing the results and providing direction for future work.
Chapter 2
Related Work

The literature review here is focused on other research done comparing the statistical performance of gene set methods. In particular, we focus on comparisons that involve using simulated gene expression data to test at least and compare at least two gene set test methods and on comparison tests that compare a large number of statistical methods.

One of the most comprehensive simulation tests was done by Wu et al. [8] comparing the statistical performance of their method, ROAST, with the GSEAlm method (a linear model version of GSEA). In their research, they simulated datasets for 10,000 genes. Each gene had its own variance which was sampled from the inverse-\(\chi^2\) distribution \(\frac{1}{\sigma^2} \sim \frac{1}{s_0^2} \chi^2_{d_0}\) where \(d_0 = 4\) and \(s_0 = 0.25\). The genes were divided into gene sets in two ways, 250 gene sets of size 40 and 10 gene sets of size 1000. A full set of test scenarios were run for each setup. Although not explicitly stated, we presume that those gene set sizes represent typical gene sets of interest in real data. For each of these two gene set setups, 10 test scenarios were tested. Each test scenario was a certain combination of the parameters: proportion of genes up-regulated in the set (0, 0.2, 0.25, 0.5, 1), proportion of genes down-regulated in the set (0, 0.2, 0.5), inter-gene correlation (0 or 0.1), and the log-fold change to be used for the DE genes (0.1, 0.2, 0.3, 0.4). The log-fold change for each scenario was chosen to meet a theoretical power given the test setup as opposed to a biologically meaningful value. The first eight scenarios were designed to test statistical power and were run with 1000 simulated data sets. In these scenarios only the first gene set contained DE genes according to the test scenario parameters. The last two scenarios were designed to test type I error rate and were run on 3000 simulated data sets. In these scenarios, no genes were generated with DE. It is again presumed, but not explicitly stated, that the 10 test scenarios represent realistic gene expression data. A final note of interest is they generated data for three groups, but
only tested two groups. For one complete set of simulations, they used $n_1 = n_2 = 3$ and $n_3 = 20$ samples and for another set, they used $n_1 = n_2 = 5$ and $n_3 = 20$ samples. The data for the third group were passed into the method, but were not included in the test for significance. In this way, ROAST likely outperformed GSEAlm because it was able to “make use of residual degrees of freedom from the 20 arrays in the third group that is not involved in the hypothesis test” [8].

Wu and Smyth [9] conducted a similar set of simulations comparing their method, CAMERA, with the methods geneSetTest, PAGE, and sigPathway. As in the ROAST test, they again used 10,000 genes and generated the data according to an inverse-$\chi^2$ distribution. In this case, the genewise variances were sampled from the distribution: $\sigma^2 \sim \frac{s_0^2 d_0}{\chi_{d_0}^2}$ with $d_0 = 4$ and $s_0 = 0.25$. This is the reciprocal of the formula used in the ROAST study, with an added coefficient in the numerator, $d_0$. The authors state here that this generates “a distribution typical of microarray experiments” [9]. There were four test scenarios described: 100% genes DE, intergroup gene correlation = 0, log-fold change = 0.05; 100% genes DE, intergroup gene correlation = 0.05, log-fold change = 0.1; 25% genes DE, intergroup gene correlation = 0, log-fold change = 0.2; and 25% genes DE, intergroup gene correlation = 0.05, log-fold change = 0.25. Each scenario was run with 1000 simulated data sets. Other than the genewise variance used to generate the expression data, no biological justification was given for any of the choices of parameter values.

Efron and Tibshriani [17] performed a simulation study comparing the GSA method with GSEA. They ran their simulations with 1000 total genes split into 50 gene sets of size 20 and $n_1 = n_2 = 50$ samples per group. The data for all genes were generated from the normal distribution: $N(0, 1)$, then a DE effect was added to certain genes in just the first gene set of group 2 (called the DE set) based on the test scenario. The 5 test scenarios were: add 0.2 to all 20 genes in the DE set; add 0.3 to the first 15 genes in the DE set; add 0.4 to the first 10 genes in the DE set; add 0.6 to the first 5 genes in the DE set; add 0.4 to the first 10 genes in the DE set and subtract 0.4 from the last 10 genes in the DE set. No reasoning was given for the choices of sample size, gene set size, or log-fold difference.
Each of the 5 test scenarios were performed with 20 simulated data sets.

Fridley et al. [20] performed a large-scale simulation study that was the most similar to this research of all the surveyed literature. In their study, they compared a handful of standard statistical methods such as the Kolmogorov-Smirnov test, Fisher’s method, Stouffer’s method, Tail strength, Global Model, and Principal Component Analysis. They tested over 2000 different scenarios. The parameters for their scenarios were: number of genes per gene set (10, 50, 100, 500), number of associated genes (varies based on number of genes in set), the effect sizes for those associations (small = 1, medium = 2, large = 3), inter-gene correlation (0, 0.1, 0.3), standard deviation of gene expression data (1, 3, 6), and sample size (20, 100, 50). For each scenario, 1000 datasets were generated and tested. Gene expression values were generated from $\text{MVN}(0, \Sigma)$ where $\Sigma$ was the covariance matrix for the test scenario. For scenarios without gene correlations, it was a diagonal matrix containing the variance of each gene. For scenarios with gene correlation, it is a structure that contains 1 for the variance on the diagonal and the correlation value determining the off-diagonal. For the differentially expressed genes, they were sampled from a normal distribution $N(\mu, \sigma)$ where $\mu = 0, 1, 2, 3$ (for no effect, small effect, medium effect, or large effect). The total number of genes used for the test scenarios was not mentioned. Again, no biological justification was given for any of the choices of parameter values.

Song and Black [22] did a simulation study comparing GSEA, GSEAIm, SAFE, GlobalTest, sigPathway, and PCOT2 (Principal Coordinates with Hotelling’s $T^2$). In this simulation study, they simulated data for 400 total genes split equally into 20 gene sets with 20 genes per gene set. There were 2 groups with 20 samples in each group. There were 4 basic scenarios which were performed under various values of inter-gene correlation and the actual mean difference between groups: both groups inactive with no significant difference, group 1 genes active and group 2 genes inactive with a significant difference, both groups active with no significant difference, and both groups active with a significant difference. They first ran each basic scenario holding the mean difference constant at 0.5 or 1 over a range of inter-gene correlation values ranging from 0 to 0.5 in increments of 0.05. Then,
they ran the basic scenarios again, this time holding the correlation constant at either 0.1 or 0.25 over a range of mean differences from 0.5 to 5 in increments of 0.5. For each test run, 100 data sets were simulated. They then used two real data sets to obtain realistic values of $p$ (total genes), and estimates for $\mu$ (mean difference) and $\Sigma$ (covariance matrix) and performed the tests again.

Goeman and Bühlmann [23] conducted a simulation study to show the relationship between the inter-gene correlation and type I error rate. They generated 5000 data sets for each value of inter-gene correlation from 0 to 1 in steps of 0.1. The data sets each had 10,000 genes which were divided into 100 gene sets with 100 genes in each. There were 2 groups with 10 samples in each group. The data were generated from a multivariate normal distribution with mean of 0 and variance of 1. They showed that as the inter-gene correlation increases, the number of false positives increases.

Liu et al. [24] performed a simulation study to compare 3 gene set test methods (Global Test, ANCOVA Global Test, and SAM-GS). In their study, they simulated data for a single gene set containing 100 genes for 2 groups of samples and tested both power and size for a combination of 2 sample sizes (10 per group and 25 per group) and 3 inter-gene correlation values (0, 0.5, and 0.9). Gene expression data was generated from a multivariate distribution. For the size (type I error rate) tests, none of the genes were differentially expressed. For the power tests, 40 of the 100 genes were differentially expressed.

A number of other papers were found that compared performance of gene set test methods but either did not use simulated data sets or were more limited in scope. Dinu et al. [25] compared 6 gene set test methods (GSEA, SAM-GS, Global Test, ANCOVA Global Test, sigPathway, and Tomfohr) on the NCI-60 data set. Tarca, Bhatti, and Romero [26] performed a comparison test of sixteen methods under relaxed assumptions while using 42 real datasets. No simulations were done in either study. Bayerlov et al. [27] compared performance of SPIA, CePa and Pathnet using simulated data sets. These methods are actually considered pathway topology-based enrichment methods, which are similar to gene set test methods but use additional information about the structure or topology of the genes.
Evangelou et al. [28] proposed that some standard association tests could be adapted to test for gene set enrichment. They compared their results only to GSEA and found the adapted association tests generally outperformed GSEA.
Chapter 3
Biological Relevance

3.1 Overview

The primary objective of this research was to compare statistical performance (power and false discovery rate) of gene set testing methods for various biologically realistic and relevant scenarios. Publicly available gene expression data sets from experiments are both realistic and relevant, but pose some interesting challenges. Since it is not known \textit{a priori} which gene sets are differentially expressed, it is not possible to determine which significant findings are true positives and which are false positives. It was also not feasible to find a large sample of data sets for each of our desired test scenarios. We therefore used simulated data for this research.

To ensure that the simulated data sets were biologically realistic and relevant, we identified a number of biological parameters of interest that would be used in the simulation framework. For each one, we performed an analysis to determine the range of values that would capture a majority of real-life scenarios. In this way, the simulation results show how each of the gene set test methods perform over a variety of realistic scenarios.

3.2 Parameters of Interest

The parameters of interest that we chose to explore and model in the simulations were: the per gene expression variance $\sigma^2$, the total number of genes $p$, the number of samples per group in experiments $n$, the number of genes in each gene set $m$, the fold-change difference in expression between groups that is actually biologically meaningful $\mu_d$, the proportion of genes in the gene set that are differentially expressed $\pi_d$, and the inter-gene correlation $r$. Other parameters, such as the degree of gene set overlap, were considered but ultimately left out of the scope of this research.
3.2.1 Sample Size

For the number of replicate samples per group, \( n \), in typical experiments, we randomly selected 60 gene expression data sets from the Gene Expression Omnibus (GEO) [29,30]. We excluded experiments which involved repeat measurements on the same subjects as well as a few experiments which had very unbalanced designs. There was no restriction on the study year, but slightly more weight in the randomization was given to studies from 2013 and newer. For each study, the average number of samples per groups was recorded and summarized.

As seen in Fig. 3.1, the distribution is right skewed with most experiments using 10 or less samples per group. 88% of the experiments used 6 or less samples per group and 95% of the experiments used 10 or less samples per group. For the number of samples per group of the 60 sampled experiments, the summary statistics are shown in Table 3.1. Further non-random investigation of GEO datasets revealed that nearly all experiments used 50 or fewer samples per group. For the purposes of our simulations, we decided to use the values \( n \in \{2, 4, 6, 8, 10, 25, 50\} \).

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>2</td>
</tr>
<tr>
<td>1st quartile</td>
<td>3</td>
</tr>
<tr>
<td>median</td>
<td>3</td>
</tr>
<tr>
<td>3rd quartile</td>
<td>5</td>
</tr>
<tr>
<td>max</td>
<td>23</td>
</tr>
<tr>
<td>mean</td>
<td>4.7</td>
</tr>
<tr>
<td>standard deviation</td>
<td>4.07</td>
</tr>
<tr>
<td>experiments with ( \leq 6 ) samples</td>
<td>88.33%</td>
</tr>
<tr>
<td>experiments with ( \leq 10 ) samples</td>
<td>95%</td>
</tr>
</tbody>
</table>

3.2.2 Gene Set Size

To estimate the distribution of gene set size parameter, \( m \), we wrote an R [21] script to traverse the human genome as annotated in the R library org.Hs.eg.db [31] and cross reference each gene set with the Gene Ontology library GO.db [32] and AnnotationDbi [33]
Fig. 3.1: Distribution of the number of samples per group in 60 randomly selected GEO data sets to determine which ones were annotated as biological processes. For each gene set that was annotated as a biological process, the size of the gene set was recorded. There were 22302 gene sets in org.Hs.eg.db, 15795 of which were tagged as biological processes in GO.db. The distribution of gene set sizes for every biological process in the human genome was extremely right-skewed with a median of 7 and a mean of 130.8807. Full summary statistics can be seen in Table 3.2. We note that the hierarchical nature of the gene sets in the human genome library (org.Hs.eg.db) creates some very large gene sets that are not biologically meaningful. For instance, the root biological process GO:0008150 contains every single

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>1</td>
</tr>
<tr>
<td>1st quartile</td>
<td>2</td>
</tr>
<tr>
<td>median</td>
<td>7</td>
</tr>
<tr>
<td>3rd quartile</td>
<td>31</td>
</tr>
<tr>
<td>max</td>
<td>38038</td>
</tr>
<tr>
<td>mean</td>
<td>130.8807</td>
</tr>
<tr>
<td>standard deviation</td>
<td>903.7878</td>
</tr>
<tr>
<td>gene sets with $\leq$ 50 genes</td>
<td>80.78%</td>
</tr>
<tr>
<td>gene sets with $\leq$ 100 genes</td>
<td>87.24%</td>
</tr>
<tr>
<td>gene sets with $\leq$ 150 genes</td>
<td>90.42%</td>
</tr>
</tbody>
</table>
gene involved in any biological process. One of its children, GO:0048518, contains any gene involved in positive regulation of biological process [4]. In the context of gene set testing, these giant generic container gene sets have no meaning. As seen in Table 3.2 and Fig. 3.2, most gene sets contain 100 or fewer genes.

Given the results of this analysis, we would have ideally used a range of gene set sizes such as from 10 to 100 in increments of 10. After exploring each of the parameters of interest, we devised a complete list of desired test scenarios. Due to the large number of test scenarios and computation time that would be needed, it was impractical to run them all. As a result, we had to scale back some of the less interesting parameters, such as gene set size. In particular, we decided to use \( m = 40 \) genes per gene set for all test scenarios.

![Cumulative density plot of the gene set size from every biological process gene set in the human genome.](image)

Fig. 3.2: Cumulative density plot of the gene set size from every biological process gene set in the human genome.
3.2.3 Genewise Expression Variance

Prior work has been done in exploring the expression distribution of each gene. Lönnstedt and Speed [34] and Wright and Simon [35] both proposed that the variance for each gene follows an inverse gamma distribution common to all genes. Smyth [36] extended this idea and suggested that gene-wise variances follow an inverse-\(\chi^2\) distribution \(\frac{1}{\sigma^2} \sim \frac{1}{d_0 s_0^2} \chi^2_{d_0}\). Further work by Wu et al. [8] and Wu and Smyth [9] also used the same inverse-\(\chi^2\) distribution and suggested that using the values \(d_0 = 4, s_0 = 0.25\) will generate realistic gene expression data sets.

3.2.4 Biologically Relevant Differential Expression

There is an important difference between statistical significance and biological importance. A result is statistically significant when the observed difference was not likely due to chance. A result is biologically meaningful when the observed difference (effect size) is large enough to be important. It is common for biological experiments to have statistically significant results that are not biologically meaningful and vice versa [37]. Therefore, we are primarily interested in the statistical power of the gene set test methods in detecting biologically meaningful differences. To determine a range of biologically meaningful differential expression values for our DE fold-change parameter, \(\mu_d\), we turn to the literature.

Yang et al. [38] suggest that fold-change values as low as 1.3 may be biologically meaningful. Huggins et al. [39] used a fold-change cutoff of 1.3 to define meaningful differential expression. Peart et al. [40] and Raouf et al. [41] both considered fold-change differences of 1.5 and above to be biologically meaningful. McCarthy and Smyth [42] used a fold-change of 1.5 and suggested that fold-change differences of 1.1 or smaller are definitely not biologically meaningful. Combining all of these results suggests that fold-change differences less than 1.1 are definitely not meaningful, 1.1 to 1.3 are probably not meaningful, 1.3 to 1.5 might be meaningful, and greater than 1.5 are most likely meaningful. We therefore used fold-change values \(\mu_d \in \{1.3, 1.5, 1.7\}\).
3.2.5 Inter-Gene Correlation

Gatti et al. [43] performed a meta-analysis on a total of 202 datasets (8,656 arrays) from the Gene Expression Omnibus [29, 30] and concluded that inter-gene correlation is widespread in gene expression data and that inter-gene correlation inflates the rate of false discoveries. From their scatterplots, it appears that most of the inter-gene correlations were between 0 and 0.2 and almost all of them were between 0 and 0.3.

Wu and Smyth [9] also showed that inter-gene correlation increases the type I error rate of gene set test methods. They computed inter-gene correlations for all gene sets with 5 or more genes in the C2 collection of the Molecular Signatures Database Version 3.0 (MSigDB) [3, 44]. The mean inter-gene correlation was 0.0026 and ranged up to 0.71. From their boxplot of correlations, it appears that there was 1 negative correlation, that the IQR of correlations was between 0 and approximately 0.15 and that almost all of the correlations were between 0 and 0.4.

For our simulations, we used the values $r \in \{0, 0.1, 0.2, 0.3\}$ for inter-gene correlation.

3.2.6 Proportion of Differentially Expressed Genes

Finally, the proportion of genes in the gene set that are differentially expressed can realistically be anything between 0 and 1. We used $\pi_d \in \{0, 0.25, 0.5, 0.75, 1\}$ to cover the full range of possible values.

3.3 Summary

We have attempted to identify important biological parameters and estimate representative ranges of values to use in our simulations. All parameters of interest are summarized here in Table 3.3. The actual test scenarios that were derived from this list and which were used in the simulations is described in the following chapter.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Justification</th>
</tr>
</thead>
</table>
| genewise expression variance $\sigma^2$       | $\frac{1}{\sigma^2} \sim \frac{1}{d_0 s_0} \chi^2_{d_0}$  
$d_0 = 4, s_0 = 0.25$ | multiple sources [8, 9, 36]      |
| total number of genes $p$                     | 2,000, 4,000, 6,000  
8,000, 10,000 | 1,000 previously used [17]  
10,000 previously used [8, 9, 23] |
| gene set size $m$                             | 40                                  | analysis of human genome [31–33] |
| samples per group $n$                         | 2, 4, 6, 8, 10, 25, 50             | sample of 60 experiments        
from GEO [29, 30]                  |
| biologically relevant fold-change $\mu_d$     | 1.3, 1.5, 1.7                       | multiple sources [38–42]        |
| proportion of DE genes $\pi_d$               | 0, 0.25, 0.5, 0.75, 1               | cover complete possible range    |
| inter-gene correlation $r$                    | 0, 0.1, 0.2, 0.3                    | Gatti et al. [43]                
Wu and Smyth [9]                     |
Chapter 4
Gene Set Test Methods

4.1 Overview

We tested a total of 19 gene set methods: mvGST, mvGST: Hartung, GlobalTest, ROAST: mean, ROAST: msq, ROAST: floormean, CAMERA, ROMER: mean, ROMER: floormean, GSEAImperm, GSA: maxmean, GSA: mean, GSA: absolute mean, sigPathway: NTk, sigPathway: NEk, SAFE: permutation, SAFE: bootstrap.t, SAFE: bootstrap.q, and PAGE (PGSEA). Rather than describe them individually, this chapter summarizes the major types of approaches, with references appearing in a concluding table.

PANTHER [13] is a popular classification system that can also do gene set testing. Ingenuity Pathway Analysis [19] is a commercial product that can do gene set testing. Both were excluded due to lack of R support. We also excluded geneSetTest [45] because it has been been superseded by CAMERA as noted by its authors [9].

4.2 Hypothesis Type

As originally stated by Goeman and Bühlmann [23], there are two main types of gene set tests, competitive and self-contained. In competitive gene set tests, each gene set $G$ is compared to the gene set consisting of all other genes not in $G$ - its complement, $G^c$. The null hypothesis for the competitive test is that the genes in $G$ are differentially expressed as often as the genes in $G^c$. Self-contained gene set tests compare the genes in a set to a fixed standard and do not use information from genes outside of the set. The null hypothesis for the self-contained is that no genes in $G$ are differentially expressed.

In general, it is thought that self-contained tests have more power than competitive tests but can be too powerful [23] and that competitive tests may be prone to serious errors
in both significance and interpretation [24]. Self-contained tests also have the ability to test a gene set consisting of all genes, whereas competitive tests cannot.

The competitive tests we tested were: CAMERA, GSA: maxmean, GSA: mean, GSA: absolute mean, PAGE (PGSEA), ROMER: mean, ROMER: floormean, SAFE: permutation, SAFE: bootstrap.t, SAFE: bootstrap.q, and sigPathway: NTk. The self-contained tests were: sigPathway: NEk, mvGST, ROAST: mean, ROAST: msq, ROAST: floormean, and GlobalTest, and GSEAlmperm.

4.3 P-value Technique

The technique used to generate the gene set p-values also varies across methods. Some of them calculate the p-value by combining the p-values from each of the genes in the gene set, others by permuting over the samples, others by permuting over the genes, and others by rotating residuals.

A few of the methods use fairly standard statistical methodology. The CAMERA method uses a two-sample t-test where the test statistic $T$ has been adjusted by the Variance Inflation Factor (VIF) to account for inter-gene correlation [9]. The PGSEA method is a parametric method that assumes a normal distribution and uses a one-sample z-test to compare each gene set mean $S_m$ to the population mean $\mu$ and standard deviation $\sigma$ from the entire data set [16]. The mvGST method uses either Stouffer’s Method or Hartung’s Method to combine p-values from individual genes into a p-value for each gene set [12]. The GlobalTest assumes a normal distribution or a $\chi^2$ distribution for small sample size by default but can also estimate the distribution by permutations [11].

A number of methods use resampling techniques to generate the unknown distribution of test statistics, then check each gene set test statistic against the distribution to generate their p-values. These can be classified further by the resampling technique. Some methods permute samples, others permute genes, others permute sample and genes, and some use bootstrapping techniques. The methods that permute over samples are: sigPathway: NEk [14], GSEAlm [7]. The only method tested here that permutes over genes is sigPathway: NTk [14]. Two methods permute over both genes and samples. The SAFE method [15]
permutes over genes to find the test statistic and then permutes over samples to calculate the p-value. The GSA methods also permute over genes and samples to generate the test statistic distribution, a process they call restandardization. The statistic distribution is then used to generate p-values for each gene set [17]. Finally, two of the SAFE methods use bootstrap sampling in their tests [46]. They generate bootstrap confidence intervals and then test each gene set statistic against the lower bound of the confidence interval. One of those methods (bootstrap.q) uses the sample $\alpha$-quantile [47] for the lower bound while the other (bootstrap.t) generates the lower bound from the t-distribution.

Finally, the ROAST methods and the ROMER methods use rotation tests to generate the gene set p-values. First, they project the data onto an orthogonalized residual space. Then, they perform many random rotations of the residuals and calculate the test statistic at each rotation. The set of test statistics generated this way is used to estimate the unknown distribution (sometimes called the null distribution) of test statistics. The observed test statistic for each gene set is compared against the null distribution to calculate its p-value. [8, 45].

4.4 Summary

The gene set method hypothesis types and p-value techniques discussed here will be used again during the analysis to assess which type of hypothesis test or p-value technique has any advantage over the others given each set of circumstances. See Table 4.1 for a summary of gene set methods along with their hypothesis type and p-value calculation technique. In total, we tested 19 methods which can be divided into 11 competitive methods and 8 self-contained methods or as 3 parametric methods, 5 rotation methods, 9 resampling methods (7 permutation and 2 bootstrap), and 2 p-value combination methods.
<table>
<thead>
<tr>
<th>Gene Set Test Method</th>
<th>Hypothesis Type</th>
<th>P-value Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAMERA</td>
<td>competitive</td>
<td>parametric</td>
</tr>
<tr>
<td></td>
<td></td>
<td>two-sample t-test [9]</td>
</tr>
<tr>
<td>GSA: maxmean</td>
<td>competitive</td>
<td>resampling</td>
</tr>
<tr>
<td>GSA: mean</td>
<td></td>
<td>permutation of genes and samples [17]</td>
</tr>
<tr>
<td>GSA: absmean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGSEA</td>
<td>competitive</td>
<td>parametric</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one-sample z-test [16]</td>
</tr>
<tr>
<td>ROMER: mean</td>
<td>competitive</td>
<td>rotation of residuals [45]</td>
</tr>
<tr>
<td>ROMER: floormean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAFE: permutation</td>
<td>competitive</td>
<td>resampling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>permutation of genes and samples [15]</td>
</tr>
<tr>
<td>SAFE: bootstrap.t</td>
<td>competitive</td>
<td>resampling</td>
</tr>
<tr>
<td>SAFE: bootstrap.q</td>
<td></td>
<td>bootstrap [46]</td>
</tr>
<tr>
<td>sigPathway: NTk</td>
<td>competitive</td>
<td>resampling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>permutation of genes [14]</td>
</tr>
<tr>
<td>sigPathway: NEk</td>
<td>self-contained</td>
<td>resampling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>permutation of samples [14]</td>
</tr>
<tr>
<td>mvGST</td>
<td>self-contained</td>
<td>combine p-values</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stouffer’s Method [12,48]</td>
</tr>
<tr>
<td>mvGST: Hartung</td>
<td>self-contained</td>
<td>combine p-values</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hartung’s Adjustment [12,49]</td>
</tr>
<tr>
<td>ROAST: mean</td>
<td>self-contained</td>
<td>rotation of residuals [8]</td>
</tr>
<tr>
<td>ROAST: msq</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROAST: floormean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GlobalTest</td>
<td>self-contained</td>
<td>parametric</td>
</tr>
<tr>
<td></td>
<td></td>
<td>logistic regression [11]</td>
</tr>
<tr>
<td>GSEAlmperm</td>
<td>self-contained</td>
<td>resampling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>permutation of samples [7]</td>
</tr>
</tbody>
</table>
Chapter 5
Simulation Framework

5.1 Overview

The simulation framework was written in the R programming language [21]. All simulation tests described in this work were run on the Ember cluster at the Center for High Performance Computing at the University of Utah (CHPC). We will describe the test scenarios, define power and false discovery rate (FDR), and then discuss the simulation framework in detail. We will then comment on some tuning parameters that were used for some of the methods, describe the computing environment, and conclude with a brief comment on performance.

5.2 Test Scenarios

The test scenarios are based on the previously defined parameters of interest and their typical values (Table 3.3). In addition to estimating the power and false discovery rate of the gene set test methods as a given parameter varied, we wanted to explore possible interactions between parameters. Given the 6 parameters we identified, there were $\binom{6}{2} = 15$ possible pairings. For each pairing, we permuted over the full combination of parameter values and tested each combination with 100 unique simulated data sets.

In order to limit the total computation time and the scope of this research, we focused our efforts on the 5 most interesting pairings as follows: number of genes with sample size, sample size with fold-change, sample size with proportion DE, sample size with correlation, and proportion DE with correlation. For each pairing, we tested all combinations of parameters and estimated power and false discovery rate for each gene set test method. In total, there were 128 test scenarios across the parameter pairings. After accounting for 28 overlapping scenarios, 100 unique test scenarios were actually tested. For scenarios with 10
or fewer samples per group, 100 unique gene expression data sets were simulated and tested with all gene set test methods. For scenarios with more than 10 samples per group, 50 data sets were simulated. The test scenarios are summarized in Table 5.1.

<table>
<thead>
<tr>
<th>Number of Genes</th>
<th>Gene Set Size</th>
<th>Sample Size</th>
<th>Fold-Change $\mu_d$</th>
<th>Proportion DE $\pi_d$</th>
<th>Correlation $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000, 4000, 6000, 8000, 10000</td>
<td>40</td>
<td>2, 4, 6, 8, 10, 25, 50</td>
<td>1.5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4000</td>
<td>40</td>
<td>2, 4, 6, 8, 10, 25, 50</td>
<td>1.3, 1.5, 1.7</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>4000</td>
<td>40</td>
<td>2, 4, 6, 8, 10, 25, 50</td>
<td>1.5</td>
<td>0.25, 0.5, 0.75, 1</td>
<td>0</td>
</tr>
<tr>
<td>4000</td>
<td>40</td>
<td>2, 4, 6, 8, 10, 25, 50</td>
<td>1.5</td>
<td>1</td>
<td>0, 0.1, 0.2, 0.3</td>
</tr>
<tr>
<td>4000</td>
<td>40</td>
<td>6</td>
<td>1.5</td>
<td>0.25, 0.5, 0.75</td>
<td>1, 0, 0.1, 0.2, 0.3</td>
</tr>
</tbody>
</table>

5.3 Significance

In gene set tests, a test of significance is performed on each gene set. The null hypothesis $H_0$ for a particular gene set states that there is no difference in gene expression between sample groups. Since there are many gene sets all being tested with the same $H_0$, the results need to be adjusted for multiple hypothesis testing in order to control the FDR. If the FDR adjusted p-value for a given gene set $p_a \leq \alpha$, there is a significant difference of gene expression between groups in that gene set and we say the genes are differentially expressed between groups. A significant result is often referred to as a discovery.

5.4 Multiple Hypothesis Correction

Some of the gene set test methods used here have an FDR adjustment built in and some of the methods do not. By default, mvGST uses a Benjamini-Yekutieli correction (BY) [50] while ROAST, CAMERA, SAFE and PGSEA all use a Benjamini-Hochberg correction (BH) [51] by default. The sigPathway methods use their own implementation of q-values for multiple hypothesis testing [14]. The GlobalTest, ROMER, GSEAlm, and GSA methods did not have an FDR adjusted p-value from the function API.
When an FDR adjusted p-value was available directly from a method, we used the default adjustment method built-in to the method function call, then used the p-value returned from the method as our FDR adjusted p-value, \( p_a \). For methods that only returned raw p-values, we manually adjusted the vector of raw p-values using the `p.adjust` method built in to R [21] with the Benjamini-Hochberg correction (BH) [51], then used the adjusted p-value from the first gene set as our \( p_a \). A summary of FDR corrections used by each method, including whether it was done manually, is show in Table 5.2.

<table>
<thead>
<tr>
<th>Gene Set Test Method</th>
<th>FDR Method</th>
<th>Manually Adjusted?</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvGST</td>
<td>BY</td>
<td>N</td>
</tr>
<tr>
<td>ROAST: mean</td>
<td>BH</td>
<td>N</td>
</tr>
<tr>
<td>ROAST: msq</td>
<td>BH</td>
<td>N</td>
</tr>
<tr>
<td>ROAST: floormean</td>
<td>BH</td>
<td>N</td>
</tr>
<tr>
<td>CAMERA</td>
<td>BH</td>
<td>N</td>
</tr>
<tr>
<td>SAFE: permutation</td>
<td>BH</td>
<td>N</td>
</tr>
<tr>
<td>SAFE: bootstrap.t</td>
<td>BH</td>
<td>N</td>
</tr>
<tr>
<td>SAFE: bootstrap.q</td>
<td>BH</td>
<td>N</td>
</tr>
<tr>
<td>PGSEA</td>
<td>BH</td>
<td>N</td>
</tr>
<tr>
<td>sigPathway: NTk</td>
<td>q-value</td>
<td>N</td>
</tr>
<tr>
<td>sigPathway: NEk</td>
<td>q-value</td>
<td>N</td>
</tr>
<tr>
<td>GlobalTest</td>
<td>BH</td>
<td>Y</td>
</tr>
<tr>
<td>ROMER: mean</td>
<td>BH</td>
<td>Y</td>
</tr>
<tr>
<td>ROMER: floormean</td>
<td>BH</td>
<td>Y</td>
</tr>
<tr>
<td>GSEAImperm</td>
<td>BH</td>
<td>Y</td>
</tr>
<tr>
<td>GSA: maxmean</td>
<td>BH</td>
<td>Y</td>
</tr>
<tr>
<td>GSA: mean</td>
<td>BH</td>
<td>Y</td>
</tr>
<tr>
<td>GSA: absmean</td>
<td>BH</td>
<td>Y</td>
</tr>
</tbody>
</table>

5.5 Power

We define the power of a given method as \( P\{\text{reject } H_0 \mid H_0 \text{ is false}\} \). We stipulate that \( H_0 \) should be false whenever there is a biologically relevant difference in gene expression between groups. The power of each method, therefore, is the proportion of biologically relevant differences that it detects as significant discoveries.

For each simulated data set, a proportion of the genes in the first gene set were generated with differentially expressed values for all of the samples in the second group. The
proportion of genes and fold-change for the differentially expressed gene set varied according to the test scenario being run. Any remaining genes in the first gene set and all genes for all other gene sets were generated from a MVN distribution as described in Section 5.7.2 such that none of the other gene sets should be differentially expressed between groups.

The power, $P$, for each method at a given test scenario is estimated as $P = \frac{n_s}{s}$ where $n_s$ is the number of significant findings for gene set 1 out of the $s$ simulated data sets. For test scenarios with 10 or fewer samples, $s = 100$, and for test scenarios with 25 or 50 samples, $s = 50$. These values were chosen to balance the trade-off between accuracy and computation time.

5.6 False Discovery Rate

We define the false discovery rate (FDR) for a method as $Q = \frac{V}{R}$ where $V$ is the number of false positives and $R$ is the total number of discoveries. In other words, it is the proportion of significant results that are incorrect.

The false discovery rate for a single data set in a single test run, $i$, was estimated as $Q_i = \frac{V_i}{R_i}$ where $V_i$ is the number of false positives (significant findings other than the first gene set) and $R_i$ is the total number of significant findings for all gene sets. The false discovery for each gene set test method at a given test scenario, $Q$, is then the average false discovery rate across all $s$ test runs, or $Q = \frac{1}{s} \sum_{i=1}^{s} Q_i$.

5.7 Framework Components

The main simulation framework (see simulation.R in Appendix B) is an R script with 4 main steps: Initial Setup, Data Generation, Gene Set Tests, and Summarize Results. For each test scenario, the Initial Setup is performed once, the Data Generation and Gene Set Tests are performed $s$ times in a loop, and the Final Summary is performed once. For each scenario, the results for each gene set test method are saved to a results CSV file.

After running all pairwise scenarios for a parameter pairing, a fifth step is performed: Final Analysis. This is a second R script (see analysis.R in Appendix B) which reads in
a series of results CSV files, manipulates the data, and generates the relevant plots. Each component will now be described in further detail.

5.7.1 Initial Setup

In the initial setup, the test parameters are read in from the command line, the gene set structures for the various methods are created, and a starting time is recorded.

5.7.2 Data Generation

A matrix of gene expression data is generated based on the test scenario parameters. Only the first gene set is generated as differentially expressed. As done in previous work [8, 9, 36], we first simulate genewise variance for each gene from the inverse-\(\chi^2\) distribution:

\[
\frac{1}{\sigma^2} \sim \frac{1}{d_0 s_0^2} \chi^2_{d_0}
\]

using the values \(d_0 = 4\) and \(s_0 = 0.25\). For each gene set, a covariance matrix \(\Sigma\) is constructed and gene expression data is simulated from a multivariate normal distribution \(MVN(\mu, \Sigma)\).

For test scenarios without inter-gene correlation \((r = 0)\), \(\Sigma\) is a diagonal matrix of the genewise variances, \(\sigma^2\). For test scenarios with inter-gene correlation \((r \neq 0)\), a correlation matrix, \(R\), is constructed containing 1 along the diagonal and the inter-gene correlation coefficient, \(r\), everywhere else. A diagonal matrix, \(D\), is also constructed from the genewise standard deviations, \(\sigma\). Then, the covariance matrix is the result of the matrix multiplication \(\Sigma = D \times R \times D\). See Appendix A for a discussion about how the method used to generate data with inter-gene correlation was validated with a feature of the CAMERA method [45] that can estimate inter-gene correlation.

The value of \(\mu\) depends on the values of the parameters \(m, \mu_d,\) and \(\pi_d\) for the test scenario currently being run. For the first sample group, \(\mu = 0\) for all genes. For the second sample group, the first \(\pi_d \times m\) genes in the first gene set have \(\mu = \log_2(\mu_d)\). For all other genes in the second group, \(\mu = 0\).

5.7.3 Gene Set Tests

The simulated data set is then tested sequentially by each of the 19 gene set test
methods. The data is massaged into the appropriate objects and formats required by the method, the method is called, and p-values are obtained for all of the gene sets. If the method returned raw p-values, they are manually converted to FDR adjusted gene set p-values as discussed in Section 5.4.

The FDR adjusted p-values for each gene set are then tested against the predefined \( \alpha \) to determine significance. If an adjusted gene set p-value, \( p_a \leq \alpha \), the result is recorded as significant for that gene set. Otherwise, it is recorded as not significant. The framework allows any value of \( \alpha \) to be specified, but the value \( \alpha = 0.05 \) was used for all test scenarios.

5.7.4 Summarize Results

After all \( s \) data sets have been generated and tested, the power and FDR for each of the 19 gene set test methods are summarized and saved to a uniquely named CSV file for future processing by the Final Analysis step. The CSV file contains a row for each method in the test and columns for the name of the test, the power, and the FDR.

The format of the filename is sim_{a,b,c,d,e,f}.csv where \( a \) is total number of genes, \( b \) is the gene set size, \( c \) is the number of samples per group, \( d \) is the fold-change value, \( e \) is the proportion of genes that are DE, and \( f \) is the inter-gene correlation coefficient. In this way, we are able to run concurrent tests on the CHPC cluster and later combine the results for analysis.

For each test completed, we also append a row to another CSV file, runlog.csv indicating the test scenario parameters used, the start and end time, and the test duration. This file was used to track CPU usage and calculate performance statistics.

5.7.5 Final Analysis

The test scenarios were designed to show both the impact of changing a parameter of interest over its distribution of values and the interaction between pairs of parameters of interest.

A series of test scenarios is the full combination of parameter values between a particular pair of parameters of interest being tested. For example, with the pair number of genes
(2000, 4000, 6000, 8000, 10000) and sample size (2, 4, 6, 8, 10), there are $5 \times 5 = 25$ total test scenarios. These scenarios were run concurrently in batches on the University of Utah CHCP cluster.

After a full series of related test scenarios was complete, the results are combined and aggregated by a post-processing R script (analysis.R) to produce the desired summary statistics and plots. The script loads every CSV in a directory and generates a new data set with the gene set test methods as rows and columns for the significance rate, the total number of genes, the gene set size, the sample size, the fold change, the proportion of genes that were differentially expressed, and the inter-gene correlation. This data was used to generate the lattice plots seen in the Simulation Results chapter.

5.8 Tuning Parameters

5.8.1 Number of Rotations

The 4 ROAST methods calculate the p-value by randomly rotating residuals in space [8]. The p-value is calculated as: $(b + 1)/(nrot + 1)$ where $b$ is the number of rotations giving a more extreme statistic than that observed [52]. This smallest possible p-value is then $1/(nrot + 1)$. For ROAST to have power, it is vital to specify enough rotations so that it is even theoretically possible to achieve a significant finding at the specified $\alpha$. By default, ROAST methods use 999 rotations. In testing, we found that with about $n_g = 125$ gene sets (5000 gene sets of 40 genes), using the default 999 rotations was insufficient to find significance most of the time when considering the FDR corrected p-value. We empirically determined that $nrot = n_g \times 20$ was sufficient to return fairly stable p-values over multiple ROAST function calls with the same data set across multiple scenarios. The simulation framework then uses $n_g \times 20$ rotations depending on the test scenario.

The ROMER methods also estimate the p-value by rotating orthogonalized residuals [10]. By default, ROMER methods use 9999 rotations. It was determined that using 9999 rotations was very CPU intensive and not practical given the total number of tests. As with ROAST, we found that using $n_g \times 20$ rotations produced fairly stable p-values that
allow ROMER to have adequate power across scenarios.

5.8.2 Number of Permutations

The sigPathway method and the GSEAlm method both use permutations to calculate the p-value. The GSA method uses permutations to estimate the FDR rate. For each of these methods, the number of permutations can be specified in the function call. By default, sigPathway uses 1000 permutations, and GSA uses 200 permutations. GSEAlm does not have a default number of permutations. In testing some possible choices for this parameter in the simulation framework, it was empirically shown that using 1000 permutations/rotations for each of these methods was sufficient to produce fairly consistent p-values across function calls with the same data set while keeping computation time fairly reasonable. The simulation framework uses 1000 permutations for each of these methods for all scenarios.

5.9 Test Environment

All simulations were performed on the Ember cluster at the CHPC. The cluster had 144 Dual Socket-Six Core Nodes (1728 total cores). Each core was a 2.8 GHz Intel Xeon (Westmere X5660) processor with 2 Gbytes RAM per core. The cluster was running the Linux CentOS release 7.4.1708 operating system.

The tests were run with R version 3.4.4 (2018-03-15) and a handful of R libraries which were installed in the user subdirectory. The installed R libraries were primarily the various gene set methods being tested. A summary of the R libraries is shown in Table 5.3.

Multiple test scenarios were run concurrently on different nodes in the cluster and the results were later combined and analyzed as previously described. The SLURM scheduler on the system controlled the queuing and scheduling of jobs.

5.10 Performance

An early version of the simulation framework did not scale well with the number of genes. As the total number of genes $p$ increased, the run time increased exponentially as
Table 5.3: List of R Libraries Used

<table>
<thead>
<tr>
<th>R Library</th>
<th>Version</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvnfast [53]</td>
<td>0.2.5</td>
<td>rmvn used to generate multivariate normal gene expression data</td>
</tr>
<tr>
<td>mvGST [54]</td>
<td>1.14.0</td>
<td>mvGST gene set test [12]</td>
</tr>
<tr>
<td>globaltest [55]</td>
<td>5.32.0</td>
<td>GlobalTest gene set test [11]</td>
</tr>
<tr>
<td>limma [45]</td>
<td>3.34.9</td>
<td>ROAST gene set tests (mean, msq, mean50, floormean) [8]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CAMERA gene set test [9]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ROMER gene set tests (mean, floormean, mean50) [10]</td>
</tr>
<tr>
<td>GSEAlm [56]</td>
<td>1.38.0</td>
<td>GSEAlmperm gene set test [7]</td>
</tr>
<tr>
<td>GSA [57]</td>
<td>1.03</td>
<td>GSA gene set tests (maxmean, mean, absmean) [17]</td>
</tr>
<tr>
<td>sigPathway [58]</td>
<td>1.46.0</td>
<td>sigPathway gene set tests (NTk and NEk) [14]</td>
</tr>
<tr>
<td>safe [59]</td>
<td>3.18</td>
<td>SAFE gene set tests (permutation, bootstrap.t and bootstrap.q) [15]</td>
</tr>
<tr>
<td>PGSEA [60]</td>
<td>1.52.0</td>
<td>PAGE gene set test [16]</td>
</tr>
</tbody>
</table>

seen in the Initial Version plot in Fig. 5.1. When testing with \( p = 10,000 \) genes, each test run was taking an average of 37.89 minutes on the CHPC cluster. This was not practical given the number of tests we were planning on running.

After profiling the code, it was determined that approximately 97.5% of the CPU time was spent generating the data while only about 2.5% of the CPU time was spent performing the actual gene set tests. In order to generate gene expression data with inter-gene correlation within gene sets, the data generation process was creating a single \( p \times p \) covariance matrix \( \Sigma \) that was being passed to the R function mvrnorm from the MASS library [61]. To increase performance, a number of optimizations were considered, and two were implemented.

The biggest gain in performance was found by replacing the single large covariance matrix with a covariance matrix for each gene set. If there are \( p \) total genes divided into \( k \) gene sets containing \( m \) genes each, there was originally a \( p \times p \) covariance matrix \( \Sigma \) that was being passed to the mvrnorm function to generate the gene expression data for each of the sample groups. When \( p \) was large (10,000), the resulting covariance matrix was found to consume hundreds of megabytes of memory. The original covariance matrix was a block diagonal matrix containing \( k \) matrices of size \( m \times m \). The matrices in the diagonal contained the genewise variance for the gene set and any inter-gene correlation. The majority of the covariance matrix contained the value 0 and had no effect on the
resulting data. The optimization made was to replace Σ with a covariance matrix for each gene set, Σₖ, call mvrnorm in a loop for each gene set, and append the generated data to the final data set. This resulted in a dramatic reduction in computation time. For \( p = 1000 \), the data generation process was more than 150 times faster than before. As \( p \) increased, the performance gain was even bigger. For \( p \geq 2500 \), the data generation process was over 1000 times faster.

The second performance optimization came from using rmvn (from mvnfast library version 0.2.5) [53] instead of mvrnorm (from MASS library version 7.3.49) [61] to generate the multivariate normal data. The performance gain was more even across different values of \( p \). On average, rmvn generated data approximately 2.3 times faster than mvrnorm.

In the original version, the average time for each run of the simulation framework increased exponentially in the number of genes \( p \), while in the optimized version, the average run time increased linearly with \( p \) as seen in Fig. 5.1. For tests with 10,000 genes, the optimized version took about 0.90 minutes per test run compared to 37.89 minutes per run originally.

![Simulation Performance](image)

Fig. 5.1: Performance Comparison of Original Version and Optimized Version
Chapter 6
Simulation Results

6.1 Overview

In addition to estimating the power and FDR of each gene set test method as a biological parameter of interest was varied over realistic and relevant values, we explored possible interactions between parameters of interest by combining parameters and running a set of simulations for each combination of parameter values. Each parameter pairing is summarized with lattice plots for power and FDR for each method, lattice plots for power and FDR for each method type, and a discussion of the results.

We devised 5 interesting pairings of our parameters of interest: total number of genes ($p$) with sample size ($n$), sample size with proportion DE ($\pi_d$), sample size with fold-change ($\mu_d$), sample size with inter-gene correlation ($r$), and proportion DE with inter-gene correlation.

For each pair of parameters, 100 unique data sets were generated (50 when $n = 25$ and $n = 50$) and tested with each gene set method and the power and FDR were recorded for each method. In total, 100 test scenarios were tested, 8,700 data sets were generated, and 165,300 gene set tests were performed. The test scenarios consumed a total of 313.04122 CPU hours on the CHPC for an average runtime of 3.1304 hours per scenario.

Many of the methods tested (mvGST, ROAST, ROMER, GSA, SAFE, and sigPathway) have multiple test statistics or algorithm that can be used that generate different results. We considered each of these variations its own gene set test method. Including all of these variations, we tested 19 different gene set test methods from the 10 main gene set test platforms. In order be able to visualize and interpret the results for the various methods, we separate the results for power and FDR into 2 plots, each showing 9 methods. The
ROAST: msq and ROAST: floormean performed similarly across test scenarios, so we chose to omit ROAST: msq from the plots to improve readability.

The following sections describes the results for the 5 chosen parameter pairings. For each pairing, we discuss trends in power and FDR individually, then compare method performance where performance is defined as achieving a certain power threshold while simultaneously controlling the FDR at $Q \leq 0.05$. For the discussion, we will consider $P \geq 80\%$ to be adequate power and $P \geq 90\%$ to be high power. Finally, we consider performance by hypothesis type (self-contained and competitive) and by p-value technique (resampling, rotation, parametric, p-value combination).

### 6.2 Number Of Genes with Sample Size

This test pairing was performed by combining each value of total number of genes, $p \in \{2000, 4000, 6000, 8000, 10000\}$, with each value of sample size, $n \in \{2, 4, 6, 8, 10, 25, 50\}$, for a total of 35 test scenarios. These scenarios were run with no inter-gene correlation, and gene sets of size 40 where all 40 of the genes in the test gene set are differentially expressed by a fold-change value of either 1.5 for power test or 0 for type I error test. The purpose of this test pair was to see how sample size affects power, how total number of genes affect power, and if there is any interaction effect between sample size and total number of genes on power.

#### 6.2.1 Power

The power of each method given each combination of total number of genes and sample size can be seen in the lattice plots in Fig. 6.1 and Fig. 6.2. The number of samples appears on the x-axis and the number of genes appears in the vertical groups (rows).

Only the GSEAlm method appears to be greatly affected by the total number of genes, $p$. As $p$ increases, the power of GSEAlm decreases. This is due to its relatively high p-values which do not hold up very well to the multiple hypothesis adjustment. The GSA absolute mean method (GSA.AM) appears to have been mildly affected by number of genes, with perhaps a slight drop in power for higher $p$. The number of genes appears to have little
effect on the power for the remaining methods with the plots looking approximately the same going from bottom to top.

The sample size, however, appears to have a dramatic effect on power for many of the methods. For \( n = 2 \), GSA maxmean (GSA.MM), GSA mean (GSA.M), mvGST, and PAGE had surprisingly high power, while the other methods had low power as expected. By increasing to \( n = 4 \) samples, the power increased substantially for all methods across all total gene sizes except for all 3 GSA methods and GSEAlm which somehow dropped. After increasing to \( n = 6 \) samples, all of the methods except GSEAlm had reasonable power.

Fig. 6.1: Method Power by Total Number of Genes and Sample Size (1 of 2)
Fig. 6.2: Method Power by Total Number of Genes and Sample Size (2 of 2)
6.2.2 FDR

The FDR of each method given each combination of total number of genes, \( p \), and sample size, \( n \), can be seen in the lattice plots in Fig. 6.3 and Fig. 6.4. A reference line has been added at our \( \alpha = 0.05 \). For FDR, the number of genes, \( p \), appears to have a mild effect on a few methods with generally slightly higher FDR for higher \( p \). This effect was compounded somewhat for smaller sample sizes.

Only CAMERA and the 2 mvGST methods correctly controlled the FDR at \( \leq 0.05 \) across all sample sizes and gene set sizes. The ROAST and ROMER methods also performed well. They controlled FDR for most scenarios and were only slightly above when they did not control it. This deviation could be attributed to random chance. Most of the other methods had fairly high FDR for small \( n \) which then decreased as \( n \) increased. For these methods, the FDR was more or less controlled at 0.05 by \( n = 6 \) or \( n = 8 \) samples. For large \( p \), the SAFE bootstrap methods both appear to need about 50 samples to control the FDR. For the permutation-based SAFE method, the FDR appears to shoot up dramatically at \( n = 4 \), then drop quite slowly after that indicating that many more samples might be needed to control the FDR. Curiously, the PAGE methods started with FDR \( \leq 0.05 \) for \( n = 2 \) which then actually increased as \( n \) increased.

6.2.3 Performance

To compare performance of the various methods, we consider power and FDR together. Table 6.1 ranks the methods according to the number of samples needed to high adequate power while controlling the FDR across all gene set sizes. The mvGST method performed the best followed closely by CAMERA and the 3 ROAST methods. These 5 methods were the only ones that uniformly controlled FDR at \( \leq 0.05 \) while also having high power at small sample sizes. Of these methods, mvGST had good power at \( n = 2 \) and the others at \( n = 4 \). The ROMER methods and GlobalTest controlled FDR at \( n \geq 4 \) and had satisfactory power somewhere around \( 4 \leq n \leq 6 \). At \( n \geq 6 \) samples, mvGST Hartung, GSA maxmean, GSA mean, and sigPathway NEk also performed well. It is worth noting that the FDR of sigPathway NEk appeared to become uncontrolled with larger samples. At \( n \geq 10 \), GSA
Fig. 6.3: Method FDR by Total Number of Genes and Sample Size (1 of 2)
Fig. 6.4: Method FDR by Total Number of Genes and Sample Size (2 of 2)
absolute mean and sigPathway NTk performed well.

Table 6.1: Samples to Achieve 90% Power and Control FDR at .05 by Method When Gene Set is 100% DE

<table>
<thead>
<tr>
<th>Gene Set Test Method</th>
<th>Sample Size Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvGST</td>
<td>2</td>
</tr>
<tr>
<td>CAMERA</td>
<td>4</td>
</tr>
<tr>
<td>ROAST: mean</td>
<td>4</td>
</tr>
<tr>
<td>ROAST: msq</td>
<td>4</td>
</tr>
<tr>
<td>ROAST: floormean</td>
<td>4</td>
</tr>
<tr>
<td>ROMER: mean</td>
<td>4</td>
</tr>
<tr>
<td>ROMER: floormean</td>
<td>4</td>
</tr>
<tr>
<td>mvGST: Hartung</td>
<td>4</td>
</tr>
<tr>
<td>GlobalTest</td>
<td>4 to 6</td>
</tr>
<tr>
<td>sigPathway: NEk</td>
<td>6</td>
</tr>
<tr>
<td>GSA: maxmean</td>
<td>6</td>
</tr>
<tr>
<td>GSA: mean</td>
<td>8</td>
</tr>
<tr>
<td>GSA: absmean</td>
<td>8</td>
</tr>
<tr>
<td>sigPathway: NTk</td>
<td>≈ 25</td>
</tr>
<tr>
<td>SAFE: bootstrap.t</td>
<td>≈ 50</td>
</tr>
<tr>
<td>SAFE: bootstrap.q</td>
<td>≈ 50</td>
</tr>
<tr>
<td>SAFE: permutation</td>
<td>≫ 50</td>
</tr>
<tr>
<td>PAGE</td>
<td>-</td>
</tr>
<tr>
<td>GSEAlm</td>
<td>-</td>
</tr>
</tbody>
</table>

Recall from Section 3.2.1 and Fig. 3.1 that many real experiments use sample sizes of \( n = 2 \) or \( n = 3 \). For these typically used sample sizes, most methods will either fail to achieve adequate power, have a very high FDR, or both. That is assuming 100% of the genes in the gene set are differentially expressed as was done in this test. With fewer DE genes, the power will be even lower. In other words, many experiments currently being done with small sample sizes may be finding many false positives while failing to find true positives. The results of this test pairing suggest that at least \( n = 6 \) samples should be the absolute minimum used in practice.

6.2.4 Performance by Hypothesis Type and P-value Technique

The average power and FDR of each hypothesis type and p-value technique is shown in Fig. 6.5 and Fig. 6.6. It appears that self-contained null hypothesis methods outperform
competitive null hypothesis methods in both power and FDR. Both types had poor power at $n = 2$, but self-contained methods generally had more power at $n \geq 4$ for any $p$. They also had lower FDR for all $n$ and all $p$.

For p-value technique, the p-value combination methods performed best overall with proper FDR control across scenarios and high power. The rotation based methods also performed well with FDR control and good power for $n \geq 4$. The resampling methods and parametric methods may have had slightly more power, but much higher FDR. The resampling methods appear to be improving with more samples while the parametric inexplicably appear to be getting worse with more samples.
Fig. 6.6: Method Type FDR by Total Number of Genes and Sample Size
6.3 Sample Size with Proportion DE

Next, we consider a perhaps more realistic case where less than 100% of the genes in the test gene set are differentially expressed. We estimate how many samples would be needed by each method to achieve a desirable power while controlling FDR for different proportions of differentially expressed genes.

In this test pairing, we hold the total number of genes at 4000, the gene set size at 40, the inter-gene correlation at 0, the fold-change at 1.5, and vary the sample size and the proportion of the test gene set that is differentially expressed. Here, we test each value of sample size, $n \in \{2, 4, 6, 8, 10, 25, 50\}$, with each value of proportion DE, $\pi_d \in \{0.25, 0.50, 0.75, 1\}$. There were a total of 28 test scenarios.

6.3.1 Power

The power of each method as a function of the sample size and the proportion of gene set that is differentially expressed is shown in the lattice plots in Fig. 6.7 and Fig. 6.8. As expected, power is low for most methods when the sample size is small or the proportion of genes differentially expressed is low. In general, the power of each method increases as either sample size increases or as the proportion of DE genes increases. There is also a definite interaction between these 2 parameters. As the proportion of genes in the gene set that are differentially expressed decreases, the methods require more samples to detect a significant difference.

When 25% of the genes are DE, only GSA mean (GSA.M) and GSA maxmean (GSA.MM) had decent power to detect significance at small sample sizes. As the sample size increases to 8 or 10, power dramatically improves for a number of methods, specifically mvGST, PAGE, ROAST mean, ROAST floormean, both variations of sigPathway, SAFE bootstrap.q, and SAFE bootstrap.t. To achieve desirable power, CAMERA, GlobalTest, and mvGST Hartung appear to need even between 10 and 25 samples. SAFE and both ROMER variations appear to need many more samples.

When 50% of the genes were DE, the power of a number of methods increases quite a bit. Specifically, mvGST, GSA maxmean, SAFE bootstrap.q, and sigPathway NEk appear
Fig. 6.7: Method Power by Proportion Differentially Expressed and Sample Size (1 of 2)
Fig. 6.8: Method Power by Proportion Differentially Expressed and Sample Size (2 of 2)
to achieve high power with a sample size of 2; mvGST Hartung, PAGE, SAFE, sigPathway NTk, and SAFE bootstrap.t by sample size of 4; CAMERA, ROAST mean, ROAST floormean, and ROAST msq by 6; GlobalTest, ROMER mean, and ROMER floormean, GSA absolute mean by 8 or 10. When most or all of the genes are DE, most methods appear to do fairly well with 4 or 6 samples.

6.3.2 FDR

The FDR rates for the various methods are shown in Fig. 6.9 and Fig. 6.10. In this test setup, the mvGST methods and CAMERA and all 3 ROAST methods more or less control the FDR at $\leq 0.05$ for all combinations of sample size and proportion DE. The ROMER methods also performed well. For a number of methods, the FDR is quite high for small sample sizes, $n \leq 4$ and decreases as $n$ increases with some methods controlling FDR by $n = 6$ and most by $n = 10$. SAFE appears to need a much larger $n$ to control the FDR. The FDR for PAGE actually appears to increase as $n$ increases.

6.3.3 Performance

Next we consider method performance in terms of power and FDR together. When 50% of the gene set is DE, mvGST outperformed all other methods, requiring only 2 samples to achieve good power and control FDR. A handful of other methods perform well with either 4 or 6 samples.

Table 6.2 ranks the methods according to the number of samples needed to achieve adequate power while controlling the FDR in the case where 25% of the gene set is DE. ROAST floormean, GSA maxmean (GSA.MM), and sigPathway: NEk performed the best, requiring 6 samples. As before, the FDR of sigPathway NEk appeared to become uncontrolled with larger samples. The mvGST method, the SAFE bootstrap methods, and the other 2 GSA methods also performed well at 8 samples.

Since knowing a priori what proportion of the gene sets will be differentially expressed is impossible, it would be best to plan for lower proportions of DE. Therefore, it would appear
Fig. 6.9: Method FDR by Proportion Differentially Expressed and Sample Size (1 of 2)
Fig. 6.10: Method FDR by Proportion Differentially Expressed and Sample Size (2 of 2)
Table 6.2: Samples to Achieve 90% Power and Control FDR at .05 by Method When Gene Set is 25% DE

<table>
<thead>
<tr>
<th>Gene Set Test Method</th>
<th>Sample Size Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROAST: floormean</td>
<td>6</td>
</tr>
<tr>
<td>GSA: maxmean</td>
<td>6</td>
</tr>
<tr>
<td>sigPathway: NEk</td>
<td>6</td>
</tr>
<tr>
<td>mvGST</td>
<td>8</td>
</tr>
<tr>
<td>GSA: mean</td>
<td>8</td>
</tr>
<tr>
<td>GSA: absmean</td>
<td>8</td>
</tr>
<tr>
<td>SAFE: bootstrap.t</td>
<td>8</td>
</tr>
<tr>
<td>SAFE: bootstrap.q</td>
<td>8</td>
</tr>
<tr>
<td>sigPathway: NTk</td>
<td>10</td>
</tr>
<tr>
<td>ROAST: mean</td>
<td>10</td>
</tr>
<tr>
<td>GlobalTest</td>
<td>10</td>
</tr>
<tr>
<td>CAMERA</td>
<td>10</td>
</tr>
<tr>
<td>ROMER: mean</td>
<td>$\gg 50$</td>
</tr>
<tr>
<td>ROMER: floormean</td>
<td>$\gg 50$</td>
</tr>
<tr>
<td>SAFE: permutation</td>
<td>$\gg 50$</td>
</tr>
<tr>
<td>PAGE</td>
<td>-</td>
</tr>
<tr>
<td>GSEAIm</td>
<td>-</td>
</tr>
</tbody>
</table>

that using at least $n = 8$ to $n = 10$ samples per group or even more would be advisable in most conditions to achieve adequate power and control FDR for most methods.

6.3.4 Performance by Type and Technique

The average power and FDR of each type and technique is shown in Fig. 6.11 and Fig. 6.12. Self-contained methods had lower FDR than competitive methods across all scenarios and better power for $n \geq 4$. When the proportion of DE genes is very low and $n = 2$, competitive methods had marginally more power than self-contained methods.

Of the 4 techniques, p-value combination techniques had the lowest FDR across scenarios and high power. For $n \geq 6$, these methods are the clear winner. The rotation-based methods also controlled FDR and appear to have had higher power for $n = 4$. The resampling methods and parametric methods had similar power profiles to the combination techniques, but both had much higher FDR. As before, FDR decreased with $n$ for resampling methods and increased for parametric methods with $n$. 
Fig. 6.11: Method Type Power by Sample Size and Proportion DE
Fig. 6.12: Method Type FDR by Sample Size and Proportion DE
6.4 Sample Size with Fold-Change

In prior research, a fold-change difference of 1.3 [38, 39] or 1.5 [40–42] have both been considered biologically relevant. In this test pairing, we hold the number of genes at 4000, the size of gene sets at 40, the inter-gene correlation at 0, the proportion of genes in the gene set that are differentially expressed at .5, and vary the sample size, \( n \in \{2, 4, 6, 8, 10, 25, 50\} \), with fold-change, \( \mu_d \in \{1.3, 1.5, 1.7\} \), for a total of 21 test scenarios.

6.4.1 Power

As expected, we see in Fig. 6.13 and Fig. 6.14 that more samples are needed to detect smaller fold-change differences in expression. There is quite a bit of variability between the methods here.

![Fig. 6.13: Method Power by Fold-Change and Sample Size (1 of 2)](image-url)
For fold-change of 1.3, the GSA maxmean and GSA mean methods detected significance fairly well with just 2 samples; mvGST and sigPathway NTk with 4 samples; SAFE, ROAST floormean, ROAST msq, and sigPathway NTk with 6 samples; mvGST Hartung, CAMERA, ROAST mean and PAGE with 8 samples; and GlobalTest and both ROMER methods appear to need somewhere between 10 and 25 samples.

As the fold-change amount is increased, fewer samples are needed for adequate power for each method. At the 1.5 fold-change difference traditionally considered as biologically relevant, several of the methods had good power with 4 samples. Other methods required 6 or 8 samples, or in the case of GlobalTest, perhaps slightly more than 10 samples to achieve adequate power.

Fig. 6.14: Method Power by Fold-Change and Sample Size (2 of 2)
At the larger 1.7 fold-change difference, most methods still do a poor job of finding significance when the sample size is small. By about 6 samples, most of the methods perform well.

6.4.2 FDR

As with the other test pairings, the FDR is quite high for some methods for small sample sizes (see Fig. 6.15 and Fig. 6.16). Most notably, the GSA methods, GSEAlm, SAFE bootstrap.q, sigPathway NEk, and, to a lesser extent GlobalTest, have very high FDR when the sample size \( n < 6 \), but become controlled around \( n = 6 \). The GlobalTest actually then fails to control FDR as \( n \) increases further.

Fig. 6.15: Method FDR by Fold-Change and Sample Size (1 of 2)
Fig. 6.16: Method FDR by Fold-Change and Sample Size (2 of 2)
6.4.3 Performance

The method performance comparison for a fold-change of 1.3 is shown in Table 6.3. Here, mvGST is the clear winner requiring just 4 samples to perform well. A handful of other methods perform well with 6 or 8 samples.

Table 6.3: Samples to Achieve 90\% Power and Control FDR at .05 by Method for Fold-Change of 1.3

<table>
<thead>
<tr>
<th>Gene Set Test Method</th>
<th>Sample Size Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvGST</td>
<td>4</td>
</tr>
<tr>
<td>ROAST: msq</td>
<td>6</td>
</tr>
<tr>
<td>ROAST: floormean</td>
<td>6</td>
</tr>
<tr>
<td>GSA: maxmean</td>
<td>6</td>
</tr>
<tr>
<td>sigPathway: NEk</td>
<td>6</td>
</tr>
<tr>
<td>mvGST: Hartung</td>
<td>6</td>
</tr>
<tr>
<td>SAFE: bootstrap.t</td>
<td>8</td>
</tr>
<tr>
<td>ROAST: mean</td>
<td>8</td>
</tr>
<tr>
<td>GSA: absmean</td>
<td>8</td>
</tr>
<tr>
<td>CAMERA</td>
<td>8</td>
</tr>
<tr>
<td>SAFE: bootstrap.q</td>
<td>10</td>
</tr>
<tr>
<td>GSA: mean</td>
<td>10</td>
</tr>
<tr>
<td>ROMER: mean</td>
<td>10 to 25</td>
</tr>
<tr>
<td>ROMER: floormean</td>
<td>10 to 25</td>
</tr>
<tr>
<td>sigPathway: NTk</td>
<td>( \approx 25 )</td>
</tr>
<tr>
<td>GlobalTest</td>
<td>( \approx 25 )</td>
</tr>
<tr>
<td>SAFE: permutation</td>
<td>( \gg 50 )</td>
</tr>
<tr>
<td>PAGE</td>
<td>-</td>
</tr>
<tr>
<td>GSEAIm</td>
<td>-</td>
</tr>
</tbody>
</table>

6.4.4 Performance by Type and Technique

The average power and FDR of each type and technique is shown in Fig. 6.17 and Fig. 6.18. To detect various fold-change differences, self-contained and competitive methods had similar power trends. Competitive methods had slightly higher power to detect a fold-change difference of 1.3 for \( n = 4 \), but self-contained methods had more power for \( n = 6 \). Both methods were similar with larger \( n \). With higher fold-change differences, the self-contained methods had higher power for all \( n \). Self-contained methods also had much lower FDR than competitive methods for all \( n \).
Fig. 6.17: Method Type Power by Sample Size and Fold-Change
Fig. 6.18: Method Type FDR by Sample Size and Fold-Change
The p-value combination methods performed the best of the method types with the lowest FDR and generally the highest power across scenarios. The rotation-based methods had slightly higher FDR, but still more or less controlled it, and slightly lower power in general. The resampling methods and parametric methods had adequate power, except for small samples, but failed to control the FDR. Once again, as \( n \) increased, the FDR for resampling methods decreased, but parametric methods increased.

### 6.5 Sample Size with Inter-Gene Correlation

In this test pairing, we consider how inter-gene correlation affects power and FDR for different sample sizes. Here, we hold the number of genes at 4000, the size of gene sets at 40, the proportion of genes in the gene set that are differentially expressed at 1, and vary the sample size, \( n \in \{2, 4, 6, 8, 10, 25, 50\} \), with inter-gene correlation, \( r \in \{0, 0.1, 0.2, 0.3\} \), for a total of 28 test scenarios.

#### 6.5.1 Power

The power plots in Fig. 6.19 and Fig. 6.20 show the power for each method as a function of inter-gene correlation (x-axis) and sample size (rows). The general trend for most of the methods is that as inter-gene correlation increases, power decreases. This is because the effective sample size is decreased with correlated data.

For smaller sample sizes \( (n \leq 4) \), the effect is quite pronounced for a number of methods. As the sample size increases, the inter-gene correlation effect becomes a lot smaller and, in many cases, appears to flatten out (mvGST Hartung, ROAST methods, ROMER methods, PAGE). A few of the methods seem to be less affected by inter-gene correlation when finding true positives. In particular, the correlation appears to have almost no effect on the power of mvGST and SAFE and very little effect on GSEAIm, both SAFE bootstrap methods, and both sigPathway methods. The plot suggests that using a sample size \( n \geq 10 \) would be advisable to mitigate the loss of effective sample size due to inter-gene correlation.
Fig. 6.19: Method Power by Samples Size and Inter-Gene Correlation (1 of 2)
Fig. 6.20: Method Power by Samples Size and Inter-Gene Correlation (2 of 2)
6.5.2 FDR

The FDR plots for this scenario are shown in Fig. 6.21 and Fig. 6.22. The methods appear to fall into 2 basic groups.

![Method FDR by Sample Size and Intergene Correlation](image)

Fig. 6.21: Method FDR by Sample Size and Intergene Correlation (1 of 2)

The first group of methods seems to be primarily unaffected by inter-gene correlation with flat FDR lines at each sample size. Of these, CAMERA and the 3 ROAST methods appear to more or less control the FDR at $\leq 0.05$ for all sample sizes and several other methods control FDR for larger sample sizes (all 3 GSA methods and GSEAlm when $n \geq 6$, GlobalTest and sigPathway NTk when $n \geq 10$). Curiously, the FDR for the SAFE method appears to be only mildly affected by the inter-gene correlation, but gets progressively worse as the sample size increases.
Fig. 6.22: Method FDR by Samples Size and Intergene Correlation (2 of 2)
The other group of methods is definitely affected by inter-gene correlation. For these methods, as the inter-gene correlation increases, the FDR increases, sometimes quite dramatically. This group includes the mvGST methods, both sigPathway methods, and both SAFE bootstrap methods. The mvGST Hartung method appears to have a downward trend indicating it might perform well for higher correlation; however, correlation values above 0.4 are unrealistic in real data as previously established in Section 3.2.5.

Increasing the sample size helps mitigate the effect of correlation on FDR for a few methods, but has no apparent effect on others. As sample size is increased, the FDR drops across correlation values for both SAFE bootstrap methods and the sigPathway NEk method, but does not seem to have much effect on FDR for the mvGST methods or sigPathway NTk.

6.5.3 Performance

The method performance comparison across all inter-gene correlation values is shown in Table 6.4. CAMERA, the GSA methods, the ROAST methods, and the ROMER methods performed best. All of them required about 8 samples to achieve 90% power and control the FDR at .05. A handful of methods did not work very well with inter-gene correlation, mainly due to their inability to control the FDR.

6.5.4 Performance by Type and Technique

The average power and FDR of each type and technique is shown in Fig. 6.23 and Fig. 6.24.

Overall, self-contained and competitive tests performed similarly with inter-gene correlation. Both types had fairly high power for \( n \geq 6 \) and near perfect power for \( n \geq 8 \). For smaller sample sizes, competitive tests had slightly more power. Neither type of method controlled FDR with nearly identical trends.

Regarding method technique, rotation methods performed best. Rotation was the only technique to control the FDR. These methods achieved good power for \( n = 6 \) or \( n = 8 \) when correlation was highest. Parametric methods had similar trends in power and FDR with
Table 6.4: Samples to Achieve 90% Power and Control FDR at .05 by Method Across Inter-Gene Correlation

<table>
<thead>
<tr>
<th>Gene Set Test Method</th>
<th>Sample Size Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAMERA</td>
<td>8</td>
</tr>
<tr>
<td>ROAST: mean</td>
<td>8</td>
</tr>
<tr>
<td>ROAST: msq</td>
<td>8</td>
</tr>
<tr>
<td>ROAST: floormean</td>
<td>8</td>
</tr>
<tr>
<td>ROMER: mean</td>
<td>8</td>
</tr>
<tr>
<td>ROMER: floormean</td>
<td>8</td>
</tr>
<tr>
<td>GSA: maxmean</td>
<td>8</td>
</tr>
<tr>
<td>GSA: mean</td>
<td>8</td>
</tr>
<tr>
<td>GSA: absmean</td>
<td>8</td>
</tr>
<tr>
<td>GlobalTest</td>
<td>10</td>
</tr>
<tr>
<td>sigPathway: NEk</td>
<td>10</td>
</tr>
<tr>
<td>mvGST</td>
<td>-</td>
</tr>
<tr>
<td>mvGST: Hartung</td>
<td>-</td>
</tr>
<tr>
<td>sigPathway: NTk</td>
<td>-</td>
</tr>
<tr>
<td>SAFE: permutation</td>
<td>-</td>
</tr>
<tr>
<td>SAFE: bootstrap.t</td>
<td>-</td>
</tr>
<tr>
<td>SAFE: bootstrap.q</td>
<td>-</td>
</tr>
<tr>
<td>PAGE</td>
<td>-</td>
</tr>
<tr>
<td>GSEAlm</td>
<td>-</td>
</tr>
</tbody>
</table>
Fig. 6.23: Method Type Power by Sample Size and Inter-Gene Correlation
Fig. 6.24: Method Type FDR by Sample Size and Inter-Gene Correlation
perhaps slightly higher power and slightly higher (and uncontrolled) FDR. The resampling methods and combination methods had good power, but struggled with FDR with increasing $r$.

6.6 Proportion DE with Inter-Gene Correlation

In this test pairing, we hold the number of genes at 4000, the size of gene sets at 40, the sample size $n = 6$, and test each value of proportion DE, $\pi_d \in \{0.25, 0.5, 0.75, 1\}$ with each value of inter-gene correlation, $r \in \{0, 0.1, 0.2, 0.3\}$, for a total of 16 test scenarios to see if the proportion DE and inter-gene correlation have an interaction effect.

6.6.1 Power

The power plots (Fig. 6.25 and Fig. 6.26) show that for most methods, the power decreases as inter-gene correlation increases across the various proportion DE, $\pi_d$, and that lower values of $\pi_d$ accelerate the negative effect of inter-gene correlation on power.

When 25% of the genes were DE, only sigPathway NTk, mvGST, and SAFE bootstrap.q held power decently well over increasing correlation, although none of them were exceptional. When 50% of the genes were DE, mvGST, sigPathway NTk, and the SAFE bootstrap methods all held power rather well over increasing correlation.

6.6.2 FDR

The FDR results for this scenario are shown in Fig. 6.27 and Fig. 6.28. A number of methods performed quite well here. CAMERA, GlobalTest, all 3 ROAST methods, and GSEAlm more or less control FDR at $\leq 0.05$ across all proportions of DE and correlation. GSA maxmean and GSA mean perform fairly well when $\pi_d \geq .75$. The FDR appears to be completely independent of the proportion of DE genes.

6.6.3 Performance

None of the methods performed well with low proportion DE and moderate to high inter-gene correlation. The methods which performed best on power over increasing corre-
Fig. 6.25: Method Power by Proportion DE and Intergene Correlation (1 of 2)
Fig. 6.26: Method Power by Proportion DE and Intergene Correlation (2 of 2)
Fig. 6.27: Method FDR by Proportion DE and Intergene Correlation (1 of 2)
Fig. 6.28: Method FDR by Proportion DE and Intergene Correlation (2 of 2)
lation (mvGST, sigPathway NTk, and the SAFE bootstrap methods) had the worst FDR performance while the methods that controlled FDR (CAMERA, GlobalTest, ROAST, GSA) had low power. There is a definite inverse relationship between power and FDR for this parameter pairing.

6.6.4 Performance by Type and Technique

The average power and FDR of each type and technique is shown in Fig. 6.29 and Fig. 6.30.

![Fig. 6.29: Method Type Power by Proportion DE and Inter-Gene Correlation](image)

Self-contained methods had slightly more power than competitive methods when the proportion of DE genes was lower and lower FDR across tests as well. Both types of methods
Fig. 6.30: Method Type FDR by Proportion DE and Inter-Gene Correlation
had quite similar performance and trends.

Rotation-based methods and parametric methods had similar performance and trends for both power and FDR. Rotation methods performed slightly better because they controlled the FDR, while parametric methods did not. Combination methods had the best power but worst FDR. Resampling methods had the next best power and next worst FDR.
Chapter 7

Conclusion

7.1 Summary

In general, the total number of genes had only a mild negative effect on power and FDR across methods. As $p$ was increased, the power for a few methods dropped due to $p$-values not surviving the multiple hypothesis correction, and for a few methods the FDR increased with $p$. These results are not new or surprising, but confirm the value of filtering out irrelevant genes or gene sets in certain experiment scenarios.

As has been previously discussed in Section 3.2.1, many real experiments are using sample sizes of 2 or 3. As we have shown in this work, this is not an adequate sample size for power or FDR. Any results from such experiments are highly questionable. To account for real issues in gene expression data such as inter-gene correlation and to find significance when less than 100% of the gene sets’ genes are differentially expressed, a minimum of 8 to 10 samples per group should be used.

Prior research [9, 43] has shown that inter-gene correlation is prevalent in microarray data, commonly ranges between 0 and 0.4, and causes problems for gene set test methods. This work confirmed that inter-gene correlation causes problems for gene set test methods. Even mild to moderate values of inter-gene correlation reduce the power or inflate the FDR (and in many cases both) for every method. Some of the methods here performed well on power and some performed well on FDR, but none performed particularly well considering both together. This is definitely an area for potential improvement.

Considering the grouped results, self-contained methods outperformed competitive methods for most scenarios. In a few cases, competitive methods had higher power, but self-contained methods generally had higher power overall. In terms of FDR control, self-contained methods were almost strictly better than competitive methods across all scenarios.
Methods that combine individual gene p-values had high power and low FDR, even when the sample size was small, the proportion of genes differentially expressed was low, or when considering a smaller fold-change difference. The only real downside of these methods was the FDR inflated quite a bit when there was inter-gene correlation. It is worth noting that the default mvGST approach uses Stouffer’s p-value combination method which assumes zero correlation. The Hartung extension assumes some non-negative correlation, which is estimated and accounted for. This explains the improved FDR control by mvGST Hartung over mvGST in the presence of inter-gene correlation.

The rotation-based methods were a little more conservative and excelled in controlling the FDR, even when there was inter-gene correlation. They also maintained decent power, if the combination of sample size and proportion of differentially expressed genes was high enough. In general, these methods performed well with 6 to 8 samples.

The power for parametric methods followed similar trends as rotation-based methods across scenarios, but was slightly higher for small sample sizes. The FDR for parametric methods was moderately higher, and generally uncontrolled.

The resampling methods had adequate power as long as the sample size was 6 or more, but struggled with FDR across all scenarios.

7.2 Future Work

There are a number of possibilities for future work in gene set testing. Future simulation studies could use other parameters of interest. One example that was neglected here is gene set overlap. In this simulation framework, as well as other previous simulation studies, we required that each gene only belonged to a single gene set. This is not a fair assumption as a search through the GO annotations shows that it is quite common for a gene to be a member of multiple gene sets. Investigation into the degree of gene set overlap and modeling that into the simulation framework might reveal some new insights into method performance on real data. Another aspect is gene set size. For simplicity and limiting the total number of scenarios, we limited the gene set size to 40 for all scenarios. It would be interesting to do a deeper analysis of gene set sizes that are biologically meaningful.
(excluding the large hierarchical container gene sets), and then do simulations modeling the common sizes. Another interesting parameter could be the variance used when generating the gene expression data to replicate the fact that the within-group variance can vary among gene expression platforms and experiments.

A possible improvement to this work would be to have more granularity of test parameter values for sample size, inter-gene correlation, and proportion DE. For example, many of our results had noticeable jumps in power between sample sizes of 2 and 4 or 4 and 6. Including sample sizes 3 and 5 may give more insight into the trends, although the overall conclusions would probably not change.

Another possibility for improvement or exploration is to investigate the parameters of interest further to better understand their distributions, then model the test scenarios to emulate the same behavior by having the parameter values come from a distribution based on a mean and variance instead of a static value. For example, the correlation between genes is not typically the same for all pairs of genes within a gene set. Assuming the true distribution of inter-gene correlations follows a normal distribution, the simulation framework would emulate this behavior. Instead of using a correlation of 0.1 for every pair of genes in the gene set, the correlation for each pair might come from a normal distribution with a mean of 0.1 and a standard deviation of .05 instead. The general idea would be to first sample from many real datasets to identify the distribution patterns and values for each of the parameters of interest, and mimic them in the simulations.

Finally, there is a genuine need to develop a new gene set test method or alternate approach that handles inter-gene correlation more effectively. The results of this work suggest that the ideal method might somehow utilize the power of combining p-values with the FDR control of rotation-based methods, although other novel approaches may work as well or better. It may be possible to extend or revise the Hartung approach to better estimate and account for inter-gene correlation to preserve better control of the FDR.
References


Appendices
Appendix A

Validation of Inter-Gene Correlation

As described in Section 5.7.2, the simulation framework can generate correlated data for genes within each gene set. It does this by first generating a correlation matrix, $R$, with 1s on the diagonal and the inter-gene correlation coefficient, $r$, on the off-diagonal. It then constructs another matrix, $D$, which is a diagonal matrix of the genewise standard deviations, $\sigma$. Finally, the covariance matrix is the result of the matrix multiplication $\Sigma = D \times R \times D$. 

To validate this method of generating correlated data, we used a feature of the CAMERA method [9] that estimates and returns a vector of inter-gene correlations for each gene set [45]. For each $r \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, we generated 250 gene sets of size 40 for 12 samples with our framework, ran the CAMERA method, and recorded the CAMERA estimate for $r$.

The distribution of CAMERA correlation estimates for the 250 gene sets at each $r$ is shown in Fig. A.1. For each $r$, the median $r$ estimated by CAMERA is approximately the same as the $r$ we used to generate the data and the upward trend also follows as expected. The variance in each boxplot can be explained by the fact that there is randomness in the actual gene expression data as it is being sampled from $MVN(0, \Sigma)$. The evidence here strongly suggests that the framework is correctly generating correlated gene expression data.
Fig. A.1: CAMERA Inter-Gene Correlation vs Framework Inter-Gene Correlation
Appendix B

R Code

B.1 Typical Experiment Size

library(ggplot2)

# semi-random exploration of typical gene expression experiments
# random samples of experimental data from the
# Gene Expression Omnibus at NCBI (GEO)
samplesPerGroup =
c(7,6,6,4,3,3,3,3,4,4,5,3,2,2,2,2,6,3,3,5,3,3,4,3,5,3,3,3,3,6,5,
  5,10,3,10,4,3,3,3,5,4,3,4,3,4,3,6,3,3,2,3,2,18,23,20,2,3,3,3,9,6,3)
length(samplesPerGroup)

# summarize the sample size variable
summary(samplesPerGroup)
mean(samplesPerGroup)
sd(samplesPerGroup)

# what proportion of the datasets had <= 6 samplesPerGroup
length(samplesPerGroup[samplesPerGroup <= 6])/length(samplesPerGroup)
# what proportion of the datasets had <= 10 samplesPerGroup
length(samplesPerGroup[samplesPerGroup <= 10])/length(samplesPerGroup)

# generate a histogram of distribution of sample sizes
qplot(samplesPerGroup,
  geom = "histogram",
  binwidth = 1,
  fill = I("black"),
  col = I("black"),
  alpha = I(.4),
  xlim = c(0, 25),
  xlab = "Number of Samples Per Group",
  ylab = "Count")

B.2 Typical Gene Set Size
library(AnnotationDbi)
library(GO.db)
library(org.Hs.eg.db)
library(ggplot2)

# for every GO ID, get the gene set size
geneSetLengths = vector(mode="numeric", length = 0)
xx1 <- as.list(org.Hs.egGO2ALLEGS)
length(xx1)

# if this gene set is a Biological Process (BP),
# add its length to the list of gene set lengths
start_time <- Sys.time()
for(i in 1:length(xx1))
{
  # GO:0008150 is the BP root - is this GO ID a descendant (IOW a BP)?
  if(is.element(names(xx1[i]), GOBPOFFSPRING$"GO:0008150"))
  {
    geneSetLengths = c(geneSetLengths, length(xx1[[i]]) )
  }
}
Sys.time() - start_time

# get some summary statistics
length(geneSetLengths)
summary(geneSetLengths)
mean(geneSetLengths)
sd(geneSetLengths)

# determine how many genesets
length(geneSetLengths[geneSetLengths < 50])/length(geneSetLengths)
length(geneSetLengths[geneSetLengths < 75])/length(geneSetLengths)
length(geneSetLengths[geneSetLengths < 100])/length(geneSetLengths)
length(geneSetLengths[geneSetLengths < 125])/length(geneSetLengths)
length(geneSetLengths[geneSetLengths < 150])/length(geneSetLengths)
length(geneSetLengths[geneSetLengths < 175])/length(geneSetLengths)
length(geneSetLengths[geneSetLengths < 200])/length(geneSetLengths)
length(geneSetLengths[geneSetLengths < 47])/length(geneSetLengths)

# generate cumulative density for number of genes 1 to 2000
numGenesInSet = vector(mode="numeric", length = 0)
percentGeneSets = vector(mode="numeric", length = 0)
for(i in 1:2000)
{
  numGenesInSet = c(numGenesInSet, i)
percentGeneSets = c(percentGeneSets,
    length(geneSetLengths[geneSetLengths < i])/length(geneSetLengths))
}

# generate a cumulative density plot of the first 50 gene set sizes
qplot(numGenesInSet[1:50], percentGeneSets[1:50],
    xlab = "Number of Genes in Gene Set",
    ylab = "Gene Sets Containing <= Number of Genes")

B.3 Simulation Framework

# Windows/DEV options
nullDevice = "Nul"
outputFileBase = "c:\school\thesis\output\sim"
genCSVOutput = FALSE

# for running on the Linux (HPC cluster)
if(Sys.info()[["sysname"]][= "Linux")
{
    .libPaths("~/R_libs")
    nullDevice = "/dev/null"
    outputFileBase = "~/output/sim"
    genCSVOutput = TRUE
}

# load libraries
library(mvnfast)
library(limma)
library(mvGST)
library(GSEAlm)
library(GSA)
library(sigPathway)
library(safe)
library(PGSEA)
library(globaltest)

# clear output log
cat("\014")
set.seed(441)

# STEP 1: INITIAL SETUP
# STEP 1A: SETUP DEFAULT VARIABLES FOR TESTS

# s0^2 = prior variance
# d0 = prior degrees of freedom
# p = total number of genes
# p0 = number genes per gene set
# n1 = number samples in reference group
# n2 = number samples in treatment group
# foldChange = the difference to be added to group 2, set 1 genes
# pUp - proportion of genes in the DE set to be
#    up-regulated in group 2 (0 to 1)
# pDown - proportion of genes in the DE set to be
#    down-regulated in group 2
# intergeneCorr - the intergene correlation coefficient (0 to 1)
# alpha - the value to use when testing significance (0 to 1)
# numRuns = number of simulations to run
# numPerms = number of permutations/rotations to use
#             for permutation/rotation methods
s0 = .25
d0 = 4
p = 4000
p0 = 40
n1 = 6
n2 = 6
foldChange = 1.5
pUp = 1
pDown = 0
intergeneCorr = 0
alpha = 0.05
numRuns = 100
numPerms = 1000
runAll = FALSE
verboseOutput = TRUE

# STEP 1B - process optional command line to override default parameters
# the order is: numRuns p p0 n foldChange pUp intergeneCorr
args <- commandArgs(trailingOnly = TRUE)
if(length(args) > 0)
    numRuns = strtoi(args[1], base = 10)
if(length(args) > 1)
    p = strtoi(args[2], base = 10)
if(length(args) > 2)
    p0 = strtoi(args[3], base = 10)
if(length(args) > 3)
    n1 = n2 = strtoi(args[4], base = 10)
if(length(args) > 4)
    foldChange = as.numeric(args[5])
if(length(args) > 5)
    pUp = as.numeric(args[6])
if(length(args) > 6)
    intergeneCorr = as.numeric(args[7])
if(length(args) > 7)
    alpha = as.numeric(args[8])

# instantiate any computed variables after command line input processed
logFoldChange = log(foldChange, 2)
numSets = p/p0
numRots = numSets * 20

# debug output
paste("Starting Simulation Framework:", Sys.time())
paste("p = ", p, ", p0 = ", p0, ", n1 = ", n1, ", n2 = ", n2, ", fold-change = ", foldChange, ", prop = ", pUp, ", corr = ",
intergeneCorr, ", alpha = ", alpha, ", numRuns = ", numRuns, sep = "")

# STEP 1C - create the gene sets

# geneSets is for mvGST, ROAST, CAMERA, GSA, romer
# a list containing the name of each gene set set1, set2, ...
# and each set contains the list of gene names in the set
geneSets = list();

# geneSetsMatrix is for GSEAIm and SAFE
# m x p matrix; m is the number of gene sets and p the number of genes
# each element contains a 1 if gene is in the gene set, 0 otherwise
geneSetsMatrix = matrix(data = 0, nrow = numSets, ncol = p)

# gsList is for sigPathway
# a list of lists where each list contains a src, title, and probes
# (the names of genes in the gene set)
gsList = list()

# smcList is for PAGE
smcList = list()

# now actually generate the various gene set structures
for(i in 1:numSets)
{
    startGene = ((i-1)*p0 + 1)
    endGene = startGene + p0 - 1
    index <- startGene:endGene;
# add this gene set to the geneSets list
geneSets[[paste('set', i, sep = "")]] = index

# add this gene set to the geneSetsMatrix
geneSetsMatrix[i, startGene:endGene] = 1

# add this gene set to the gsList
gsListItem = list()
gsListItem$src = toString(i)
gsListItem$title = paste("set", i)
gsListItem$probes = sapply(startGene:endGene, toString)
gsList[[i]] = gsListItem

# add this gene set to the smcList
smcList[[i]] = new("smc", ids = startGene:endGene, reference = paste("simple smc", i))
}

# add gene set names to avoid annoying warning from SAFE
rownames(geneSetsMatrix) = paste("set", 1:numSets)
# convert gsList to the real gslist for sigPathway
gsList <- selectGeneSets(gsList, sapply(1:p, toString), 10, 1000)

# create roundUp function to round UP on all .5!
roundUp = function(x) trunc(x+0.5);

# function to convert 2 sided pvalues to 1 sided pvalues
p2.p1 <- function(p, diff)
{
  p1 <- rep(NA, length(p))
  t <- diff >=0
  p1[t] <- p[t]/2
  p1[!t] <- 1-p[!t]/2
  return(p1)
}

# function to perform a multiple hypothesis adjustment # to a vector of one-sided pvalues
p.adjust.onesided <- function (pvals, method = "BH")
{
  two.sided <- convertPvalues(pvals, two.sided = FALSE)
  two.adjusted <- p.adjust(two.sided, method = method)
}
relative <- ifelse(pvals < 0.5, 1, -1)
one.adjusted <- convertPvalues(two.adjusted, relative)
return(one.adjusted)

# function to get the fdr of a set of gene set p-values
get.fdr <- function(p, alpha)
{
denom = sum(p <= alpha)
if(denom == 0)
  return(0)
else
  return(sum(p[2:length(p)] <= alpha) / denom)
}

# wrapper function for mvGST methods
mvGST <- function(y, design, geneSets, corrAdjust)
{
  # fit a linear model and get 1 sided pvalues for each gene
  fit <- lmFit(y, design)
  fit <- eBayes(fit)

  fullTable = topTable(fit, coef = 2, sort = "none", n = Inf)
pvals = p2.p1(fullTable$P.Value, fullTable[,1])

  gn <- rownames(fullTable)
names(pvals) <- gn
pvals = cbind(pvals)

  # combine gene set pvalues with mvGST profileTable
  test <- profileTable(pvals,
                        organism = "hsapiens",
                        list.groups = geneSets,
                        corr.adj = corrAdjust)

  # return the BY adjusted p-values
  return(test$adjusted.group.pvals)
}

# wrapper function for GlobalTest method
GlobalTest <- function(y, design, numSets, p0)
{
  cf = as.factor(design[,2])
pvalues = c()
for(i in 1:numSets)
startGene = ((i-1)*p0 + 1)
endGene = startGene + p0 - 1

# need samples in rows and genes in columns:
mat = t(y[startGene:endGene,])
gt.all = gt(cf ~ mat)
pvalue = gt.all$result[1]
pvalues = c(pvalues, pvalue)

# return the BH adjusted p-values
return(p.adjust.onesided(pvalues, "BH"))

# RunTest - run a single parameterized test run
# simulate a data set, test all methods
# s0^2 = prior variance
# d0 = prior degrees of freedom
# p = total number of genes
# p0 = number genes per gene set
# n1 = number samples in reference group
# n2 = number samples in treatment group
# pUp = number 0 to 1 representing proportion of genes in
# the DE set to generate as up-regulated
# pDown = number 0 to 1 representing proportion of genes in
# the DE set to generate as down-regulated
# intergeneCorr = number 0 to 1 representing the intergene correlation
# logFoldChange = the log fold change to simulate for DE genes
RunTest <- function(s0, d0, p, p0, n1, n2, pUp, pDown, intergeneCorr, logFoldChange)
{
  # STEP 2 - DATA GENERATION
  # populate group means according to the requested ratio
  muGroup1 = rep(0, p0)
  muGroup2 = rep(0, p0)

  # make the first genes up, down, or both
  # according to the requested proportions
  numUp = roundUp(pUp * p0);
  numDown = roundUp(pDown * p0);
  muGroup2[1:numUp] = logFoldChange
  if(numDown > 0)
    muGroup2[numUp + 1:numDown] = -logFoldChange

  # generate data for each gene set
for(j in 1:numSets)
{
    # generate variance for each gene from inverse chi^2
    variance = (d0 * s0^2) / rchisq(p0, d0)

    # create a covariance matrix
    S = matrix(nrow = p0, ncol = p0, data = 0)
    if(intergeneCorr == 0)
    {
        diag(S) = variance
    } else
    {
        # create matrix with standard deviations on diagonal
        stdevs = sqrt(variance)
        D = diag(stdevs)

        # create correlation matrix
        R = matrix(nrow = p0, ncol = p0, data=c(rep(intergeneCorr, p0 * p0)))
        diag(R) = 1
        #print(R)

        # create covariance matrix
        S = D %*% R %*% D
    }

    # now generate and append some data
    if(j == 1)
    {
        y = rmvn(n1, muGroup1, S)
        y = t(rbind(y, rmvn(n2, muGroup2, S)))
    } else
    {
        y = rbind(y, t(rmvn(n1+n2, muGroup1, S)))
    }
}

# create the design matrix to be used by most methods
design <- cbind(Intercept=1, Group2=c(rep(0,n1), rep(1,n2)))

# construct an ExpressionSet object from our generated data
# for the GSEAlm and PAGE methods
eset = ExpressionSet(y,
                      new("AnnotatedDataFrame",
                            data = data.frame(design)))

# STEP 3 - GENE SET TESTS
# call each test and store the results

# create a data frame to hold the results from each method
results = data.frame(row.names = c("test","pvalue", "fdr"));

# GSA and SAFE both print debug messages that clog up the output file
# suppressing that here
sink(nullDevice)

# ROAST - mean
result = mroast(y, geneSets, design, contrast = 2,
    set.statistic = "mean", adjust.method = "BH",
    nrot = numRots, sort = "none")
results = rbind(results,
    data.frame(test = "ROAST.mean",
        pvalue = result$FDR[1],
        fdr = get.fdr(result$FDR, alpha)))

# ROAST - msq
result = mroast(y, geneSets, design, contrast = 2,
    set.statistic = "msq", adjust.method = "BH",
    nrot = numRots, sort = "none")
results = rbind(results,
    data.frame(test = "ROAST.msq",
        pvalue = result$FDR[1],
        fdr = get.fdr(result$FDR, alpha)))

# ROAST - mean50
#result = mroast(y, geneSets, design, contrast = 2,
#    set.statistic = "mean50", adjust.method = "BH",
#    nrot = numRots, sort = "none")
#results = rbind(results,
#    data.frame(test = "ROAST.mean50",
#        pvalue = result$FDR[1],
#        fdr = get.fdr(result$FDR, alpha)))

# ROAST - floormean
result = mroast(y, geneSets, design, contrast = 2,
    set.statistic = "floormean", adjust.method = "BH",
    nrot = numRots, sort = "none")
results = rbind(results,
    data.frame(test = "ROAST.floormean",
        pvalue = result$FDR[1],
        fdr = get.fdr(result$FDR, alpha)))
# CAMERA
result = camera(y, geneSets, design, contrast = 2,
              inter.gene.cor = NA, sort = FALSE)
results = rbind(results,
data.frame(test = "CAMERA",
pvalue = result$FDR[1],
fdr = get.fdr(result$FDR, alpha)))

# mvGST - Stouffer
pvalues = mvGST(y, design, geneSets, "none")
results = rbind(results,
data.frame(test = "mvGST",
pvalue = pvalues[1],
fdr = get.fdr(pvalues, alpha)))

# mvGST - Hartung
pvalues = mvGST(y, design, geneSets, "hartung")
results = rbind(results,
data.frame(test = "mvGST.Hartung",
pvalue = pvalues[1],
fdr = get.fdr(pvalues, alpha)))

# GlobalTest
pvalues = GlobalTest(y, design, numSets, p0)
results = rbind(results,
data.frame(test = "GlobalTest",
pvalue = pvalues[1],
fdr = get.fdr(pvalues, alpha)))

# GSEAlm
result = gsealmPerm(eset, formula = ~Group2, mat = geneSetsMatrix,
nperm = numPerms, removeShift = FALSE,
detailed = FALSE)
# FDR adjusted with BH correction
pvalues = as.numeric(p.adjust.onesided(result[,2], "BH"))
results = rbind(results,
data.frame(test = "GSEAlm",
pvalue = pvalues[1],
fdr = get.fdr(pvalues, alpha)))
# GSA - maxmean
result = GSA(y, design[,2] + 1, geneSets, 1:p, method = "maxmean",
             resp.type = "Two class unpaired",
             minsize = 1, maxsize = 1000, nperms = numPerms)
# FDR adjusted with BH correction
pvalues = p.adjust.onesided(result$pvalues.hi, "BH")
results = rbind(results,
                 data.frame(test = "GSA.MM",
                            pvalue = pvalues[1],
                            fdr = get.fdr(pvalues, alpha)))

# GSA - mean
result = GSA(y, design[,2] + 1, geneSets, 1:p, method = "mean",
             resp.type = "Two class unpaired",
             minsize = 1, maxsize = 1000, nperms = numPerms)
# FDR adjusted with BH correction
pvalues = p.adjust.onesided(result$pvalues.hi, "BH")
results = rbind(results,
                 data.frame(test = "GSA.M",
                            pvalue = pvalues[1],
                            fdr = get.fdr(pvalues, alpha)))

# GSA - absmean
result = GSA(y, design[,2] + 1, geneSets, 1:p, method = "absmean",
             resp.type = "Two class unpaired",
             minsize = 1, maxsize = 1000, nperms = numPerms)
# FDR adjusted with BH correction
pvalues = p.adjust.onesided(result$pvalues.hi, "BH")
results = rbind(results,
                 data.frame(test = "GSA.AM",
                            pvalue = pvalues[1],
                            fdr = get.fdr(pvalues, alpha)))

# romer - mean
result = romer(y, geneSets, design, contrast = 2,
                nrot = numRots, set.statistic = "mean")
# FDR adjusted with BH correction
pvalues = as.numeric(p.adjust.onesided(result[,2], "BH"))
results = rbind(results,
                 data.frame(test = "ROMER.mean",
                            pvalue = pvalues[1],
                            fdr = get.fdr(pvalues, alpha)))
# romer - floormean
result = romer(y, geneSets, design, contrast = 2,
nrot = numRots, set.statistic = "floormean")
# FDR adjusted with BH correction
pvalue = as.numeric(p.adjust.onesided(result[,2], "BH"))
results = rbind(results,
data.frame(test = "ROMER.floormean",
pvalue = pvalues[1],
fdr = get.fdr(pvalues, alpha)))

# romer - mean50
#result = romer(y, geneSets, design, contrast = 2,
nrot = numRots, set.statistic = "mean50")
# FDR adjusted with BH correction
#pvalues = as.numeric(p.adjust.onesided(result[,2], "BH"))
#results = rbind(results,
data.frame(test = "ROMER.mean50",
pvalue = pvalues[1],
fdr = get.fdr(pvalues, alpha)))

# sigPathway - NTk
result = calculate.NTk(y, design[,2], gsList, nsim = numPerms,
ngroups = 2, verbose = FALSE,
alwaysUseRandomPerm = TRUE)
results = rbind(results,
data.frame(test = "sigPathway.NTk",
pvalue = result$q.value[1],
fdr = get.fdr(result$q.value, alpha)))

# sigPathway - NEk
result = calculate.NEk(y, design[,2], gsList, nsim = numPerms,
ngroups = 2, verbose = FALSE,
alwaysUseRandomPerm = TRUE)
results = rbind(results,
data.frame(test = "sigPathway.NEk",
pvalue = result$q.value[1],
fdr = get.fdr(result$q.value, alpha)))

# SAFE - permutation
result = safe(y, design[,2], C.mat = t(geneSetsMatrix),
method = "permutation", error = "FDR.BH",
```
print.it = FALSE)
results = rbind(results,
data.frame(test = "SAFE",
pvalue = result@global.pval[[1]],
fdr = get.fdr(result@global.pval, alpha)))

# SAFE - bootstrap.t
result = safe(y, design[,2], C.mat = t(geneSetsMatrix),
method = "bootstrap.t", error = "FDR.BH",
print.it = FALSE)
results = rbind(results,
data.frame(test = "SAFE.BOOT.T",
pvalue = result@global.pval[[1]],
fdr = get.fdr(result@global.pval, alpha)))

# SAFE - bootstrap.q
result = safe(y, design[,2], C.mat = t(geneSetsMatrix),
method = "bootstrap.q", error = "FDR.BH",
print.it = FALSE)
results = rbind(results,
data.frame(test = "SAFE.BOOT.Q",
pvalue = result@global.pval[[1]],
fdr = get.fdr(result@global.pval, alpha)))

# PAGE (PGSEA)
pgNF = PGSEA(eset, smcList, range = c(10, 1000), ref = 1:n1,
p.value = NA, weighted = FALSE, enforceRange = TRUE)
fit = lmFit(pgNF, design)
fit = eBayes(fit)
pvalues = topTable(fit, sort = "none", n = numSets)$adj.P.Val
results = rbind(results,
data.frame(test = "PAGE",
pvalue = pvalues[1],
fdr = get.fdr(pvalues, alpha)))

# restoring debug printing
sink()

# print(results)
return(results);
RepeatTest <- function(s0, d0, p, p0, n1, n2, pUp, pDown, intergeneCorr, logFoldChange, numRuns, alpha)
{
    # NOTE the order of tests here must match the order in RunTest function
    # the correct code to look it up by name was too slow on performance
    test = c("ROAST.mean", "ROAST.msq", "ROAST.floormean", "CAMERA",
             "mvGST", "mvGST.Hartung", "GlobalTest", "GSEA1m", "GSA.MM",
             "GSA.M", "GSA.AM", "ROMER.mean", "ROMER.floormean",
             "sigPathway.NTk", "sigPathway.NEk", "SAFE", "SAFE.BOOT.T",
             "SAFE.BOOT.Q", "PAGE")
    numTests = length(test)
    power = rep(0, numTests)
    fdr = rep(0, numTests)
    fullResults = data.frame(test, power, fdr);

    for(i in 1:numRuns)
    {
        # run the simulation and to all tests
        results = RunTest(s0, d0, p, p0, n1, n2, pUp, pDown, intergeneCorr, logFoldChange)
        if( verboseOutput == TRUE)
            print(paste("Test Run", i, "finished:", date()))

        # look at the results and decide if significant or not
        for (row in 1:nrow(results))
        {
            # test against the predefined alpha for significance
            if( results[row, "pvalue"] <= alpha)
            {
                fullResults[row, "power"] = fullResults[row, "power"] + 1
            }

            # update the fdr rate cumulative
            fullResults[row, "fdr"] = fullResults[row, "fdr"] +
                                      results[row, "fdr"]
        }
    }

    # update the significance rate and false discovery rate for each test
    fullResults$power = fullResults$power / numRuns
    fullResults$fdr = fullResults$fdr / numRuns

    # STEP 4 - SUMMARIZE RESULTS
    print("Test Scenario finished:", quote = FALSE)
    print("s0, d0, p, p0, n1, n2, pUp, pDn, r, lfc, runs, alpha", quote = FALSE)
output = c(s0, d0, p, p0, n1, n2, pUp, pDown, intergeneCorr,
            logFoldChange, numRuns, alpha)
print(paste(output, collapse = "", " "), quote = FALSE)
print(fullResults)

# write the results to a unique file name
# based on the test scenario parameters
if(genCSVOutput)
    write.csv(fullResults,
    file = paste(paste(outputFileBase, p, p0, n1, foldChange,
          pUp, intergeneCorr, sep = "_"),
    "csv", sep = "."))

    return(1);
}

# Run the test scenario the specified number of times
startTime = Sys.time()
RepeatTest(s0, d0, p, p0, n1, n2, pUp, pDown, intergeneCorr,
          logFoldChange, numRuns, alpha);
endTime = Sys.time()
runTime = difftime(endTime, startTime, units = "mins")
runTime

# add this test run to the runlog table
runTime = as.double(runTime)
if(genCSVOutput)
{
    write.table(data.frame(p, p0, n1, n2, pUp, pDown, intergeneCorr,
            foldChange, logFoldChange, numRuns,
            startTime, endTime, runTime),
    "runlog.csv",
    col.names = !file.exists("runlog.csv"),
    row.names = FALSE,
    quote = FALSE,
    sep = ",",
    append = TRUE)
}

B.4 Simulation Analysis

library(lattice)
library(latticeExtra)
library(gdata)
# remove the results csv files - they will be regenerated
file.remove("results.csv")
file.remove("results_grouped.csv")

# methodSet = which set of methods to plot
# 1 = plot the main 9 methods for the thesis chapter
# 2 = plot the other 9 methods for the appendix
# 3 = plot by method groupings:
#   competitive, self-contained, resampling, rotation, parametric
for(methodSet in 1:3)
{

    # testPairing = which test scenario pairing to summarize:
    # 1 = numGenes/numSamples
    # 2 = numSamples/proportionDE
    # 3 = numSamples/foldChange
    # 4 = numSamples/igCorrelation
    # 5 = proportionDE/igCorrelation
    for(testPairing in 1:5)
    {

        # filename and directory stuff
        if(testPairing == 1) { pairingName = "genes_samples"} else
        if(testPairing == 2) { pairingName = "samples_proportion"} else
        if(testPairing == 3) { pairingName = "samples_foldchange"} else
        if(testPairing == 4) { pairingName = "samples_correlation"} else
        if(testPairing == 5) { pairingName = "proportion_correlation"}
        directory = paste(pairingName, "/", sep = "")

        # the file names look like: sim_a_b_c_d_e_f.csv
        # where the items are the parameters of interest
        # a = total number of genes
        # b = gene set size
        # c = number of samples per group
        # d = fold-change value
        # e = proportion of genes that are DE
        # f = inter-gene correlation coefficient

        # loop through all csv files in the directory and build a
        # new data frame with a column of significance rates for
        # each test scenario, plus columns a through f above
        results = data.frame()
        fileNames = dir(directory, pattern =".csv")
        for(i in 1:length(fileNames))
        {
            # read in the file
result = read.table(paste(directory, fileNames[i], sep = ""),
    header=TRUE,
    sep = ",",
    stringsAsFactors = FALSE)

# parse the file name into the test scenario parameters
parameters = unlist(strsplit(gsub("\.csv",
    "",
    fileNames[i],
    fixed = TRUE),
    "_"))

# add the columns for the parameters from the test
result["numGenes"] = strtoi(parameters[2], 10)
result["setSize"] = strtoi(parameters[3], 10)
result["numSamples"] = strtoi(parameters[4], 10)
result["foldChange"] = as.numeric(parameters[5])
result["proportionDE"] = as.numeric(parameters[6])
result["interGeneCorrelation"] = as.numeric(parameters[7])

# append this result to the full results
results = rbind(results, result)
}

# cleanup column names/values for plotting
results$X = NULL
colnames(results)[colnames(results) == "test"] <- "method"
results$method[results$method == "camera"] = "CAMERA"

# take a subset of the methods for plotting
csvResultsFile = "results.csv"
if(methodSet == 1) {
    results = results[results$method %in% c("mvGST",
        "ROAST.mean",
        "CAMERA",
        "GlobalTest",
        "SAFE",
        "GSA.MM",
        "sigPathway.NTk",
        "PAGE",
        "ROMER.mean"),]
    fileExtension = ".main"
} else if(methodSet == 2) {
    results = results[results$method %in% c("mvGST.Hartung",
        "ROAST.floormean",
        "GSEA.lm",
        "GSA.MM",
        "sigPathway.NTk",
        "PAGE",
        "ROMER.mean")]
    fileExtension = ".main"
}
fileExtension = "_appendix"

} else {
    # create a new data frame with
    # aggregate data for each method type:
    # self-contained, competitive,
    # rotation, resampling, parametric, combine

    # aggregate data for all competitive methods
    temp = results[results$method %in% c("CAMERA",
            "GSA.MM",
            "GSA.M",
            "GSA.AM",
            "PAGE",
            "ROMER.mean",
            "ROMER.floormean",
            "SAFE",
            "SAFE.BOOT.T",
            "SAFE.BOOT.Q",
            "sigPathway.NEk"),
            fileExtension = "_appendix"
        ]

dfp = aggregate(power ~ numGenes + setSize + numSamples +
                  foldChange + proportionDE +
                  interGeneCorrelation,
                  FUN = mean, data = temp)

dff = aggregate(fdr ~ numGenes + setSize + numSamples +
                foldChange + proportionDE +
                interGeneCorrelation,
                FUN = mean, data = temp)

results2 = cbind(method = "Competitive", merge(dfp,dff))

# aggregate data for all self-contained methods
temp = results[results$method %in% c("mvGST",
            "mvGST.Hartung",
            "ROAST.mean",
            "ROAST.msq",
            "ROAST.floormean",
            "GlobalTest",
            "sigPathway.NEk"),
            fileExtension = "_appendix"
        ]

dfp = aggregate(power ~ numGenes + setSize + numSamples +
                foldChange + proportionDE +
                interGeneCorrelation,
FUN = mean, data = temp

dff = aggregate(fdr ~ numGenes + setSize + numSamples +
    foldChange + proportionDE +
    interGeneCorrelation,
    FUN = mean, data = temp)

results2 = rbind(results2,
    cbind(method = "Self-Contained", merge(dfp,dff)))

# aggregate data for all resampling methods
temp = results[results$method %in% c("SAFE",
    "SAFE.BOOT.T",
    "SAFE.BOOT.Q",
    "GSA.MM",
    "GSA.M",
    "GSA.AM",
    "sigPathway.NTk",
    "sigPathway.NEk"),]

dfp = aggregate(power ~ numGenes + setSize + numSamples +
    foldChange + proportionDE +
    interGeneCorrelation,
    FUN = mean, data = temp)

dff = aggregate(fdr ~ numGenes + setSize + numSamples +
    foldChange + proportionDE +
    interGeneCorrelation,
    FUN = mean, data = temp)

results2 = rbind(results2,
    cbind(method = "Resampling", merge(dfp,dff)))

# aggregate data for all rotation methods
temp = results[results$method %in% c("ROAST.mean",
    "ROAST.msq",
    "ROAST.floormean",
    "ROMER.mean",
    "ROMER.floormean"),]

dfp = aggregate(power ~ numGenes + setSize + numSamples +
    foldChange + proportionDE +
    interGeneCorrelation,
    FUN = mean, data = temp)

dff = aggregate(fdr ~ numGenes + setSize + numSamples +
    foldChange + proportionDE +
    interGeneCorrelation,
    FUN = mean, data = temp)

results2 = rbind(results2,
    cbind(method = "Rotation", merge(dfp,dff)))

# aggregate data for all parametric methods
temp = results[results$method %in% c("CAMERA",
    "GlobalTest",
    "PAGE"),]
dfp = aggregate(power ~ numGenes + setSize + numSamples +
    foldChange + proportionDE +
    interGeneCorrelation,
    FUN = mean, data = temp)
dff = aggregate(fdr ~ numGenes + setSize + numSamples +
    foldChange + proportionDE +
    interGeneCorrelation,
    FUN = mean, data = temp)
results2 = rbind(results2,
    cbind(method = "Parametric", merge(dfp,dff)))

# aggregate data for all p-value combination methods
temp = results[results$method %in% c("mvGST",
    "mvGST.Hartung"),]
dfp = aggregate(power ~ numGenes + setSize + numSamples +
    foldChange + proportionDE +
    interGeneCorrelation,
    FUN = mean, data = temp)
dff = aggregate(fdr ~ numGenes + setSize + numSamples +
    foldChange + proportionDE +
    interGeneCorrelation,
    FUN = mean, data = temp)
results2 = rbind(results2,
    cbind(method = "Combination", merge(dfp,dff)))

# copy results2 into the object that will be used for the plots
results = results2

# file name options
csvResultsFile = "results_grouped.csv"
fileExtension = ".grouped"
}

# save the results table to the master file
write.table(results,
    csvResultsFile,
    col.names = !file.exists(csvResultsFile),
    row.names = FALSE,
    quote = FALSE,
    sep = ",",
    append = TRUE)

# build the plot(s) based on the test pairing being run
if(testPairing == 1)
{
    powerPlot = useOuterStrips(xyplot(power~as.factor(numSamples) | method*as.factor(numGenes),
        main = "Method Power by Number of Genes and Sample Size",
        xlab = "Number of Samples",
        ylab = "Power",
        data = results,
        type = c("a","p"))

    fdrPlot = useOuterStrips(xyplot(fdr~as.factor(numSamples) | method*as.factor(numGenes),
        main = "Method False Discovery Rate by Number of Genes and Sample Size",
        xlab = "Number of Samples",
        ylab = "False Discovery Rate",
        data = results,
        type = c("a","p"),
        panel = function(x,y,...){
            panel.refline(h = 0.05, col = "darkgrey")
            panel.xyplot(x,y,...)})
}
else if (testPairing == 2) {
    powerPlot = useOuterStrips(xyplot(power~as.factor(numSamples) | method*as.factor(proportionDE),
        main = "Method Power by Proportion DE and Sample Size",
        xlab = "Number of Samples",
        ylab = "Power",
        data = results,
        type = c("a","p"))

    fdrPlot = useOuterStrips(xyplot(fdr~as.factor(numSamples) | method*as.factor(proportionDE),
        main = "Method False Discovery Rate by Proportion DE and Sample Size",
        xlab = "Number of Samples",
        ylab = "False Discovery Rate",
        data = results,
        type = c("a","p"),
        panel = function(x,y,...){
            panel.refline(h = 0.05, col = "darkgrey")
            panel.xyplot(x,y,...)})
}
else if (testPairing == 3) {
    powerPlot = useOuterStrips(xyplot(power~as.factor(numSamples) | method*as.factor(foldChange),
        main = "Method Power by Fold-Change and Sample Size",
        xlab = "Number of Samples",
        ylab = "Power",
        data = results,
        type = c("a","p"),
        panel = function(x,y,...){
            panel.refline(h = 0.05, col = "darkgrey")
            panel.xyplot(x,y,...)})
}
xlab = "Number of Samples",
ylab = "Power",
data = results,
type = c("a","p"))

fdrPlot = useOuterStrips(xyplot(fdr~as.factor(numSamples)|
method*as.factor(foldChange),
main = "Method False Discovery Rate
by Fold-Change and Sample Size",
xlab = "Number of Samples",
ylab = "False Discovery Rate",
data = results,
type = c("a","p"),
panel = function(x,y,...){
  panel.refline(h = 0.05, col = "darkgrey")
  panel.xyplot(x,y,...)})}
}

} else if (testPairing == 4) {
  powerPlot = useOuterStrips(xyplot(power~
    as.factor(interGeneCorrelation)|
    method*as.factor(numSamples),
main = "Method Power
by Sample Size and Intergene Correlation",
xlab = "Intergene Correlation",
ylab = "Power",
data = results,
type = c("a","p"))

fdrPlot = useOuterStrips(xyplot(fdr~
  as.factor(interGeneCorrelation)|
  method*as.factor(numSamples),
main = "Method False Discovery Rate
by Sample Size and Intergene Correlation",
xlab = "Intergene Correlation",
ylab = "False Discovery Rate",
data = results,
type = c("a","p"),
panel = function(x,y,...){
  panel.refline(h = 0.05, col = "darkgrey")
  panel.xyplot(x,y,...)})
}

} else if (testPairing == 5) {
  powerPlot = useOuterStrips(xyplot(power~
    as.factor(interGeneCorrelation)|
    method*as.factor(proportionDE),
main = "Method Power
by Proportion DE and Intergene Correlation",
xlab = "Intergene Correlation",}
ylab = "Power",
data = results,
type = c("a","p"))

fdrPlot = useOuterStrips(xyplot(fdr~
                  as.factor(interGeneCorrelation)|
                  method*as.factor(proportionDE),
                  main = "Method False Discovery Rate
                  by Proportion DE and Intergene Correlation",
                  xlab = "Intergene Correlation",
                  ylab = "False Discovery Rate",
                  data = results,
                  type = c("a","p"),
                  panel = function(x,y,...){
                    panel.refline(h = 0.05, col = "darkgrey")
                    panel.xyplot(x,y,...)})

} # save the power plot to a file
png(filename = paste("..\LaTeX\images\", pairingName,
                  fileExtension,
                  ".png",
                  sep = ""),
                  width = 1200, height = 1085)
plot(powerPlot)
dev.off()

# save the fdr plot to a file
png(filename = paste("..\LaTeX\images\", pairingName,
                  "_fdr",
                  fileExtension,
                  ".png",
                  sep = ""),
                  width = 1200, height = 1085)
plot(fdrPlot)
dev.off()