5-2015

Annotation Tools for Multivariate Gene Set Testing of Non-Model Organisms

Russell K. Banks
Utah State University

ANNOTATION TOOLS FOR MULTIVARIATE GENE SET TESTING OF NON-MODEL ORGANISMS

by

Russell K Banks

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

of

MASTER OF SCIENCE

in

Statistics

Approved:

Dr. John R. Stevens Major Professor	Dr. Daniel C. Coster Committee Member
Dr. Guifang Fu Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2015

Abstract

Annotation Tools for Multivariate Gene Set Testing of Non-Model Organisms

by

Russell K Banks, Master of Science

Utah State University, 2015

Major Professor: Dr. John R. Stevens
Department: Mathematics and Statistics

Many researchers across a wide range of disciplines have turned to gene expression analysis to aid in predicting and understanding biological outcomes and mechanisms. Because genes are known to work in a dependent manner, it's common for researchers to first group genes in biologically meaningful sets and then test each gene set for differential expression. Comparisons are made across different treatment/condition groups.

The meta-analytic method for testing differential activity of gene sets, termed multivariate gene set testing (mvGST), will be used to provide context for two persistent and problematic issues in gene set testing. These are: 1) gathering organism specific annotation for non-model organisms and 2) handling gene annotation ambiguities.

The primary purpose of this thesis is to explore different gene annotation gathering methods in the building of gene set lists and to address the problem of gene annotation ambiguity. Using an example study, three different annotation gathering methods are proposed to construct GO gene set lists. These lists are directly compared, as are the subsequent results from mvGST analysis. In a separate study, an optimization algorithm is proposed as a solution for handling gene annotation ambiguities.

(93 pages)

Public Abstract

Annotation Tools for Multivariate Gene Set Testing of Non-Model Organisms

by

Russell K Banks, Master of Science

Utah State University, 2015

Major Professor: Dr. John R. Stevens
Department: Mathematics and Statistics

Microarray chip technology enables researchers to obtain measures of gene activity for essentially all genes in an organism. After grouping genes into biologically meaningful sets, researchers employ certain statistical tests to identify which gene sets (biological processes) show different levels of activity across different treatment groups. The idea is to identify which biological processes are significantly affected by a certain treatment/condition in a given organism.

Non-model organisms (such as sheep) are not widely studied so gene set membership information is not always readily accessible. This thesis work utilizes two microarray studies involving sheep to provide researchers with working examples of three different methods for gathering gene set membership information for genes in non-model organisms.

Often after gathering gene set membership information for non-model organisms, there exits ambiguity as to which set each gene belongs. A procedure for working through these ambiguities is presented. All R code used to produce the presented results is included as an appendix.

To my wife Lauren, completing this thesis is as much your accomplishment as it is mine.

Acknowledgments

Thank you Dr. John Stevens for your help on this project during the past year. You've taught me so much as an adviser and instructor and have made my time at Utah State an enjoyable and invaluable experience.

Russell K Banks

Contents

List of Tables

List of Figures

Chapter 1

Background

1.1 Introduction

The abundance of experimental gene expression measurements and annotation information has provided both opportunities and challenges for the bio-statistical community. Many sophisticated statistical methods for the analysis of high-throughput gene expression data have been developed and refined. Unsatisfyingly, different methods still tend to identify different gene sets as significantly differentially expressed with no consensus as to which method is preferable. Some of the more popular methods include GSEA, GSA, Global Ancova, SAFE, and Global testing [1]. For the purposes of this thesis, the meta-analytic method for testing differential activity of gene sets, termed multivariate gene set testing (mvGST), will be used to provide context for two persistent and problematic issues in gene set testing. These are: 1) gathering organism-specific annotation for non-model organisms and 2) handling gene annotation ambiguities.

The primary purpose of this thesis is to explore different gene annotation gathering methods in the building of gene set lists and to address the problem of gene annotation ambiguity. Using an example study, three different annotation gathering methods are proposed to construct GO gene set lists. These lists are directly compared, as are the subsequent results from mvGST analysis. In a separate study, an algorithmic approach borrowed from the field of graph theory is proposed as a solution for handling gene annotation ambiguities.

Chapter 1 of this thesis introduces previously-presented ideas and issues that are used in subsequent chapters.

1.2 Gene Expression and Gene Expression Technology

Current methods for measuring gene expression are based on the concept that DNA

codes for RNA which in turn has a molecular function within the organism or codes for polypeptides (the building blocks for proteins) [2]. In either case, the functional workings of an organism can be profiled by measures of RNA abundance and in theory is also a measure of RNA activity or gene expression activity. While it's true that every cell in an organism contains the entire genetic material specific to that organism, not every cell's DNA is expressed identically. For example, the same segment of DNA that codes for identical gene products might be active in one cell type while simultaneously being completely inert in another.

Different high-throughput technologies allow essentially all genes in the genome to be measured experimentally. These large scale experimental gene expression profiles can be obtained for thousands of genes simultaneously [3]. Generally, different gene expression measurement technologies can be classified as array-based (microarrays) or sequence-based (next-generation sequencing). There are advantages and disadvantages to both types, but such discrepancies are not immediately relevant to the purposes of this thesis. The example studies chosen for this thesis are array-based gene expression studies [4] [5].

1.2.1 Microarray Technology

The following is a physical description of a microarray. These concepts are important in considering methods to measure annotation strength, see Chapter 3.2.3.

> A DNA microarray is an orderly arrangement of thousands of identified sequenced genes printed on an impermeable solid support, usually glass, silicon chips or nylon membrane. Each attached and identified sequenced gene corresponds to a fragment of genomic DNA, cDNAs, PCR products or chemically synthesized oligonucleotides of up to 70mers and represents a single gene. Usually a single DNA microarray slide/chip may contain thousands of spots with each spot representing a single gene and collectively the entire genome of an organism [6].

These spots are annotated with specific labels called probe identifiers, or probeIDs. Each probeID in theory should be annotated to one gene identifier, or geneID. When probeIDs and geneIDs are not in a one-to-one relationship, these annotations can be called

ambiguous annotations. Microarray manufacturers often provide probeID-to-geneID annotation. An example of manufacturer mappings is given in [5], the use of which is detailed in Chapters 3.1.3 and 3.2.1. The probe sequences of the Agilent-019921 microarray are 60-mer sequences.

1.3 Preprocessing Overview

A lengthy discussion of statistical preprocessing methodology need not be discussed here. It is sufficient to understand the purpose and data structure resulting from such methods. An excellent review of the topic can be found in Zhijin Wu's paper [7].

Preprocessing of oligonucleotide arrays typically include image processing, background adjustment, data normalization/transformation and sometimes summarization when multiple probes are used to target one genomic unit [7].

The original hybridization data obtained from a microarray chip is an image. Through image processing, pixels are measured resulting in raw intensity values for each probe. Background adjustment methods remove local artifacts and "noise." Normalization methods are meant to transform the data in such a way as to make measurements from different arrays comparable in statistical analyses. Summarization methods combine probe intensity levels resulting in one gene expression level value for every probe or probe set identifier [7] [8].

While many different methods of preprocessing exist, they mostly differ in the details of how they accomplish the largely universal steps described above [7] [8]. It has also been shown that preprocessing methods can significantly impact subsequent statistical analyses [9]. RMA preprocessing (Robust Multi-Array Average) was the preprocessing method of choice in both example studies considered in this thesis [4] [5].

1.3.1 Expression Set Matrix

The results of RMA preprocessing of raw data can be summarized in an expression set matrix. The values of an expression set matrix are considered gene expression level data and are typically on the log_2 scale. Rows are specified with probeIDs and columns

specify array sample identity. An expression set matrix can be used in statistical tests of significance meant to identify differentially expressed genes.

A header of the resulting data structure after RMA preprocessing of the raw data for example study 1 is shown in table 1.4. This data was obtained through the Gene Expression Omnibus repository archived under accession number GSE47776 [4].

Table 1.1: Partial Expression Set Matrix

	GSM1159567	GSM1159568	GSM1159569	GSM1159570
AFFX-BioB-3_at	7.486	7.565	7.575	7.658
AFFX-BioB-5_at	7.393	7.365	7.277	7.357
AFFX-BioB-M_at	7.977	7.958	7.953	7.990
AFFX-BioC-3_at	9.206	9.225	9.169	9.194
AFFX-BioC-5_at	8.570	8.634	8.537	8.573
AFFX-BioDn-3_at	11.263	11.363	11.309	11.295
AFFX-BioDn-5_at	9.985	10.142	9.970	10.025
AFFX-Bt-A00196-1_s_at	3.743	3.526	3.839	4.015

Note: Full dimensions are 24128 rows by 12 columns. Data are RMA processed values from Study 1 2.1. Rows contain probeIDs of Affymetrix Bovine Microarray. Columns contain sample identifiers.

1.4 Statistical Tests for Differential Expression of Genes

There are various approaches to test for differential expression (hereafter, DE) among genes across conditions of interest for microarray experiments. Many papers compare different methods, but a general consensus as to the best has not been reached. The development of new methods is an active area of research [10] [11]. Generally, "One may distinguish between parametric tests. . . and non-parametric tests. One or two group t-test comparisons, multiple group ANOVA, and more general trend tests are all instances of linear models that are frequently used for assessing differential gene expression" [12]. Specific tests commonly used in identification of DE genes in microarray analyses include SAM (Significance Analysis of Microarrays) and Limma (Linear Models for Microarray Data) [13] [14]. Both are readily accessible in the Bioconductor project in R [15] [16]. The Limma approach was selected for the identification of DE genes in both example studies described in Chapters 2.1 and 2.2.

1.4.1 Limma/eBayes

There are a multitude of methods to test for DE among genes in microarray analysis. One commonly used method is Linear Models for Microarray Data (Limma).

> Limma is a package for differential expression analysis of data arising from microarray experiments. The package is designed to analyze complex experiments involving comparisons between many RNA targets simultaneously... The central idea is to fit a linear model to the expression data for each gene. Empirical Bayes and other shrinkage methods are used to borrow information across genes making the analyses stable even for experiments with small number of arrays [14].

Limma is flexible with respect to platform and experimental design. While use of Limma in this thesis is restricted to microarray data, the methods can be implemented for count data (RNA-Seq data for example) as if they were microarray data by way of a voom transformation as described by Charity Law et al. [17]. Multiple contrasts of interest can be tested by the Limma method. These contrasts are constructed to address researchers' hypotheses (tested contrasts for the example studies can be found in Chapters 2.1 and 2.2).

The Limma package also provides options to account for multiple tests of DE among genes. Genome-wide microarray experiments typically have thousands of probe sets representing genes, so it becomes necessary to adjust for the high number of tests to control error rates. One common method to control the false discovery rate is by comparing the Benjamini-Hochberg adjusted p-values to some rejection threshold α instead of raw p-values. Table 1.2 provides an example of output from a Limma procedure on the first contrast of example Study 1.

Table 1.2: Limma/eBayes Results with Bejamini-Hochberg Adjusted P-values

	Day12.P-Day12.NP	AveExpr	F	P.Value	adj.P.Val
AFFX-BioB-3_at	−0.025	7.744	0.034	0.857	0.984
AFFX-BioB-5_at	0.071	7.456	0.479	0.502	0.927
AFFX-BioB-M_at	0.061	8.150	0.228	0.641	0.958
AFFX-BioC-3_at	0.077	9.351	0.491	0.496	0.926
AFFX-BioC-5_at	0.028	8.722	0.069	0.797	0.979
AFFX-BioDn-3_at	0.026	11.433	0.059	0.811	0.981

Note: Only the first 6 rows of 24128 are shown. For more detail about the contrasts considered in Example Study 1 see Chapter 2.1. Rows contain probeIDs of Affymetrix Bovine Microarray. Columns contain sample identifiers.

1.4.2 Matrix of P-Values

Often researchers are interested in multiple contrasts of interest simultaneously. To test the multiple contrasts of interest for both example studies, implementation of the R package mvGST will be used. This method is described in more detail in Chapter 1.5.2.

Chapter 1.4.1 briefly describes the Limma procedure for obtaining p-values for genes in a single contrast of interest. If the p-value for a given gene is lower than some rejection threshold α we say that that gene is differentially expressed for the corresponding contrast (e.g. these genes are differentially expressed in Treatment A vs. Control). Results for tests of differential expression among genes under multiple contrasts of interested are easily summarized in a matrix of p-values, where columns indicate the contrasts tested and the rows specify probe or gene identifiers. Table 1.3 is a header of the p-value matrix obtained in example Study 1.

Table 1.3: P-Value Matrix Header

	Day12.P-Day12.NP	Day14.P-Day14.NP
AFFX-BioB-3_at	0.857	0.661
AFFX-BioB-5_at	0.502	0.527
AFFX-BioB-M_at	0.641	0.576
AFFX-BioC-3_at	0.496	0.786
AFFX-BioC-5_at	0.797	0.621
AFFX-BioDn-3_at	0.811	0.846

Note: P-values result from DE testing methods described in Chapter 1.4.1.
The full matrix contains 24128 rows.

1.5 Statistical Tests for Differential Expression of Gene Sets

Functional aspects of an organism can include gene products from numerous genes. The identification of DE genes across treatments alone proves to be inadequate for researchers concerned with deriving large-scale biological meaning due to treatment effect.

Statistical methods that test whether a gene set is differentially expressed (or differentially active) across treatment groups can yield insights into large-scale biological functions if the gene sets are constructed in biologically meaningful ways. "The study of gene set function most commonly makes use of controlled vocabulary in the form of ontology annotations" [18]. One widely used public repository where such ontology annotations can be found is provided by the Gene Ontology Consortium [19]. More about the GO Consortium is found in Chapter 1.5.1. GO annotations are the only gene-set annotations considered in this thesis for reasons discussed in Chapter 1.5.2. Exploring different methods for collecting GO annotations to build GO gene set lists is one of the primary purposes of this thesis and is demonstrated in Chapter 3.

As was the case with tests for DE among genes, there are many and varied statistical methods for testing DE among gene sets. Some of the most popular methods have been stated previously in the introduction, Chapter 1.1. The multivariate gene set testing method (mvGST) has been selected to analyze DE among different GO gene set lists, see Chapter 1.5.2 for more information on mvGST.

1.5.1 The GO Consortium

An introduction to the GO Consortium's purpose and contributing member databases can be found on the organization's web page. It is sufficient here to state, "[t]he Gene Ontology Consortium (GOC) is a set of model organism and protein databases and biological research communities actively involved in the development and application of the Gene Ontology" [19]. Prior to the GO consortium, it was all but impossible to compare functional gene category profiles across different organisms because different organisms were annotated using different conventions [3].

The GO is comprised of a collection of GO terms each of which has a unique identifier (GO ID). Each GO ID can be thought of as a list of genes where each gene has a geneID, or probeID if the microarray is directly annotated to GO. Each gene annotated to a GO term only contributes to the gene product attribute identified by the GO term. It's important to note, mvGST analysis is only appropriate for gene set lists where elements of each gene set are contributing members only.

Additionally, the GO can be thought of as a collection of three separate ontologies or domains for gene product properties. A short description of each is included below [19].

> *Cellular Component (CC): These terms describe a component of a cell that is part of a larger object, such as an anatomical structure or a gene product group.*
>
> *Molecular Function (MF): Molecular function terms describe activities that occur at the molecular level, such as "catalytic activity" or "binding activity." GO molecular function terms represent activities rather than the entities (molecules or complexes) that perform the actions, and do not specify where, when, or in what context the action takes place. Molecular functions generally correspond to activities that can be performed by individual gene products, but some activities are performed by assembled complexes of gene products.*
>
> *Biological Process (BP): A biological process term describes a series of events accomplished by one or more organized assemblies of molecular functions. The general rule*

to assist in distinguishing between a biological process and a molecular function is that a process must have more than one distinct step.

"A GO annotation consists of a GO term associated with a specific reference that describes the work or analysis upon which the association between a specific GO term and gene product is based. Each annotation also must include an evidence code to indicate how the annotation to a particular term is supported" [19]. These evidence codes are generally separated by two types: Experimental Evidence codes and Computational Analysis Evidence codes. For more information about evidence codes, one should access the GO Consortium's web page under "guide to GO evidence codes." They are not used in the annotation gathering methods contained in this thesis, but may provide additional information to measure annotation strength, see Chapter 4.3.

1.5.2 Multivariate Gene-Set Testing

The following is taken from the mvGST package help file in R and provides an introduction to the package.

> mvGST provides platform-independent tools to identify GO terms (gene sets) that are differentially active (up or down) in multiple contrasts of interest. Given a matrix of one-sided p-values (rows for genes, columns for contrasts), mvGST uses meta-analytic methods to combine p-values for all genes annotated to each gene set, and then classify each gene set as being significantly more active (1), less active (-1), or not significantly differentially active (0) in each contrast of interest. With multiple contrasts of interest, each gene set is assigned to a profile (across contrasts) of differential activity. Tools are also provided for visualizing (in a GO graph) the gene sets classified to a given profile [20].

The testing and profiling of each gene set as more active (1), less active (-1), or not significantly differentially active (0), is quite efficient and informative. Many methods for testing gene set enrichment identify only significant differential activity across a single contrast of interest let alone three levels of significant differential activity across multiple contrasts. Detailed information about the statistical methodology behind mvGST can be found in [20]. It is worthy to further emphasize here that the additional levels of differential

activity identified by mvGST stems from the understanding that every gene annotated to a GO term contributes to the functional property described by the GO term. Therefore, one-sided p-values for each gene, obtained by any statistical method for testing DE among genes, can be used as a measure of contribution to the GO term.

A Benjamani-Yekutieli adjustment [20] for multiple possible dependent comparisons will be used in mvGST analysis. It would be inappropriate to adjust for multiple comparisons twice, so raw p-values will be used to construct the matrices of p-values used in mvGST analysis, see Chapter 3.

The package mvGST is not only a tool by which DE analysis for gene sets can be conducted, it's also a tool the user can use to gather annotation information and build GO gene set lists. Annotation gathering methods provided by mvGST currently supports annotation for 22 species. This limitation is due to the fact that there are only 22 organism-specific annotation packages in Bioconductor.

These packages can contain probeID-to-GO annotation and geneID-to-GO annotation. Translation options are provided when probeID-to-GO annotations are not available but geneID-to-GO annotations are. Translation methods provided in mvGST give statistical solutions to handle gene translation ambiguities, but have the potential to bias subsequent gene set testing by omitting or creating p-values. This thesis proposes an alternative algorithmic solution for handling translation ambiguities.

1.6 Annotation Sources

An annotation in functional gene expression studies refers to a statement that a gene product has a particular functional property. These statements are stored in structured semantic databases of which the GO is an example. There are many bio-ontological databases that describe important aspects of biological domains and are often specific to field of study. These databases not only store annotation statements but also simplify or omit less important or irrelevant aspects, allowing for specific and meaningful queries [3].

In recent years, access to regularly updated bio-ontological databases has grown both more important and efficient. The increasing rate of annotation entries to databases by

biological researchers and the improvements to database user-interfaces have greatly accelerated research endeavors in the field of bioinformatics and has made access to current annotation vitally important.

1.6.1 Gene Expression Omnibus

The example studies chosen to demonstrate methods proposed in this thesis were obtained through the Gene Expression Omnibus. "[The] GEO is an international public repository that archives and freely distributes microarray, next-generation sequencing, and other forms of high-throughput functional genomics data submitted by the research community [21]." GEO records are organized by platform, sample and series.

"A Platform record is composed of a summary description of the array or sequencer and, for array-based Platforms, a data table defining the array template. Each Platform record is assigned a unique and stable GEO accession number (GPLxxx). A Platform may reference many Samples that have been submitted by multiple submitters" [21]. Submitted platform annotation information can come from the platform manufacturers, the research group (submitters), or a combination of these. We explore this topic through two specific examples in Chapter 2.

"A Sample record describes the conditions under which an individual Sample was handled, the manipulations it underwent, and the abundance measurement of each element derived from it. Each Sample record is assigned a unique and stable GEO accession number (GSMxxx). A Sample entity must reference only one Platform and may be included in multiple Series [21]."

"A Series record links together a group of related Samples and provides a focal point and description of the whole study. Each Series record is assigned a unique and stable GEO accession number (GSExxx) [21]." Information in a GSE submission include: summary of experiment, overall design, contributor(s), submission details and researchers contact information, platform information, curated data, and raw data. Both Study 1 and Study 2 are GSE records, see Chapter 2.

The Bioconductor package GEOquery provides a user interface to the GEO repository

from R. To upload information from the GEO repository into R, one need only reference the targeted experiment's GSE accession. More on GEOquery can be found in the GEOquery help file.

1.6.2 Platform Manufacturer's Annotation

The manufacturers of different gene expression technologies often provide access to annotation information that maps manufacturer designed genes to common public domain sequences [22]. For well-annotated microarray chips, direct annotation to GO might be available. This option is attractive for researchers interested in GO gene set analysis as mapping from probeIDs to GO-annotated geneIDs is unnecessary, thus avoiding researcher created mapping ambiguities (annotation ambiguities are discussed in more detail in Chapter 3.

Platform manufacturers' gene annotation can be accessed through online annotation repositories (see Chapters 3.1.2 and 3.2.1), annotation packages in R (see Chapter 3.1.1), or can be downloaded directly from the manufacturer's website (see Chapters 3.1.3 and 3.2.1). Annotation information from manufacturers' websites is stored as individual files for each array type and is frequently updated, usually quarterly [23] [22]. These files tend to be quite large, ranging from 5MB to 170 MB.

1.6.3 Ensembl's BioMart

"Ensembl is a genomic interpretation system providing the most up-to-date annotations, querying tools and access methods for chordates and key model organisms. Ensembl does not produce genome assemblies, instead [they] provide annotation on genome assemblies that have been deposited into the INSDC (GenBank, ENA, DDBJ) and are publicly available" [24].

Biomart is considered a "powerful" tool for microarray analysis [25]. The entire repository is easily accessed in R through the R package biomaRt. The annotation found in biomaRt is commonly used to interpret and map gene identifiers [25]. The current Ensembl

Release 80 has gene annotation for 69 organisms. This number is far larger than the 30 organism-specific annotation packages currently in R, see Chapter 1.6.4.

Updates and Releases

The Ensembl release cycle is approximately every three months and may occasionally be longer if considerable development work is being undertaken. Previous releases are archived as completely separate websites allowing researchers to reproduce analysis results. The current version as of May 2015 is release 80. Ensembl genes 80 is the version used in all Biomart queries of this thesis, see Chapter 3.

Previous Biomart releases can be readily accessed using the package biomaRt in R. The recommended method for accessing previous releases of Biomart is by changing the `host` parameter in the `useMart` function. Table 1.4 shows the currently available Biomart release archives.

Table 1.4: Available Biomart Release Versions and Dates

Version	Date	Version	Date
Ensemble 80	May 2015	Ensemble 70	Jan 2013
Ensemble 79	Mar 2015	Ensemble 69	Oct 2012
Ensemble 78	Dec 2014	Ensemble 68	Jul 2012
Ensemble 77	Oct 2014	Ensemble 67	May 2012
Ensemble 76	Aug 2014	Ensemble 59	Aug 2010
Ensemble 74	Dec 2013	Ensemble 54	May 2009
Ensemble 73	Sep 2013		
Ensemble 72	Jun 2013		
Ensemble 71	Apr 2013		

Biomart Query Structure

It may be beneficial to visualize an example of a Biomart query. Table 1.5 shows all Ensembl geneIDs annotated to at least one GO ID for the *Ovis aries* (sheep) data set in Study 1, see Chapter 2.2. The table also provides a third column identifying the GO domain for each geneID-to-GO annotation. Tables such as these were used to build GO gene set

lists in Chapter 3 It is interesting to note that *Ovis aries* is not an organism supported by mvGST annotation methods because there is not an org.db package for *Ovis aries* within R.

Table 1.5: Partial Results of a Biomart Query

	ensembl_gene_id	go_id	go_domain
1	ENSOARG00000000006	GO:0005739	cellular_component
2	ENSOARG00000000006	GO:0005743	cellular_component
3	ENSOARG00000000006	GO:0008137	molecular_function
4	ENSOARG00000000006	GO:0016020	cellular_component
5	ENSOARG00000000006	GO:0016021	cellular_component
6	ENSOARG00000000006	GO:0016491	molecular_function
7	ENSOARG00000000006	GO:0055114	biological_process
8	ENSOARG00000000006	GO:0070469	cellular_component

1.6.4 Annotation Packages in R

OrganismDbi is a meta framework for annotation packages in R. "The organismDbi is a software package that helps tie together different annotation resources [26]." These packages are in effect meant to be the place where "ALL" genetic information about a specific organism can be found and queried [26]. Consequently, these packages can be very large. OrgDb packages are updated quarterly. Organism-level packages in R are primarily based on mapping using Entrez Gene identifiers [26]. As of May 2015, Bioconductor provides some type of annotation information for only 30 different organisms [27].

Platform-level rather than organism-level packages are assembled packages in R using microarray platform annotation information. This information provides annotation primarily based on mapping using the platform probeIDs. For a well-annotated platform, packages that contain direct GO annotations from the GO Consortium might be available.

R tools have been developed to combine information across different annotation packages using the four methods columns, key types, keys and select [26]. If a platform package exists for a given organism but is not included in the org.db packages, they can be combined quickly and easily for the construction of gene set lists. This method would be applicable if

a platform annotation package doesn't include GO annotation. See Chapter 3.3 as to why these methods of annotation were not included in this thesis.

1.7 Stable Marriage Problem Algorithm

A stable marriage problem (SMP) algorithm accomplishes the task of finding stable mappings between elements of two different sets given input about preferences for every element in both sets [28]. The sets can be thought of as two different columns, men and women, with the elements of column one being different men and the elements of column two being different women. A mapping can be thought of as a marriage. What is meant by stability is that no marriage pairing or mapping will be better for both individuals in any given pairing. In other words, taking into account individual marriage preferences for both men and women, a SMP algorithm assures there is no pairing in which both the man and the woman would be happier. The result is a bijective mapping.

An unequal number of men and women in the two columns do not invalidate the proof, provided by David Gale and Lloyd Shapley in 1962. Their proof states that a SMP can always be solved in such a way as all marriages are stable [29]. When columns are of unequal length, the returned bijective mapping will simply be of length equal to the smallest column. The following is a somewhat simple example in that only 5 men and 4 women are considered. The example is explained according to the implementation of the SMP algorithm in the matchingMarkets R package [30].

Given 5 men and 4 women one wants to find the most stable marriages possible accounting for preferences each man and each woman provides. Let the men's preferences be expressed as the matrix $men.prefs$ with the ith column containing man i's ranking over the women in decreasing order of preference (i.e. most preferred first).

$$men.prefs = \begin{pmatrix} 1 & 4 & 1 & 3 & 2 \\ 4 & 3 & 2 & 2 & 3 \\ 3 & 2 & 3 & 1 & 4 \\ 2 & 1 & 4 & 4 & 1 \end{pmatrix}$$

.

Let the women's preferences be expressed as the matrix $women.prefs$ with the jth column containing woman j's ranking over the men in decreasing order of preference (i.e. most preferred first).

$$women.prefs = \begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & 3 & 2 & 4 \\ 3 & 1 & 1 & 3 \\ 5 & 4 & 4 & 2 \\ 4 & 5 & 5 & 1 \end{pmatrix}$$

The resulting matrix $stable.marriages$ (produced using the `daa` function from the matchingMarkets [30] package in R) provides the optimal pairings with columns representing the women and rows representing the men. Each TRUE value represents a marriage. Notice only one TRUE in each i, j element of the matrix. Row 4 (man 4) has no marriage mapping (remember there were 5 men and 4 women).

$$stable.marriages = \begin{pmatrix} TRUE & FALSE & FALSE & FALSE \\ FALSE & FALSE & TRUE & FALSE \\ FALSE & TRUE & FALSE & FALSE \\ FALSE & FALSE & FALSE & FALSE \\ FALSE & FALSE & FALSE & TRUE \end{pmatrix}$$

The SMP algorithm is applied to gene annotations within each GO term in Chapter 3.2.4.

Chapter 2

Motivation

The primary purpose of this thesis is to explore gene annotation in the building of gene set lists and address the problem of gene annotation ambiguity in mvGST analyses. The current R package mvGST has limited annotation capabilities and provides temporary methods for handling annotation ambiguities. Current methods and limitations of gathering annotations by mvGST are discussed in Chapter 1.5.2.

To conduct mvGST analysis, one must have an experiment's matrix of p-values (see Chapter 1.3) and a list of meaningful gene sets or groupings. The matrix of p-values used by the meta-analytical methods in mvGST are platform-independent, but the process of building gene set lists is not. Two example studies are discussed in this thesis. Study 1 was selected to compare mvGST results under current mvGST annotation methods to mvGST results under annotation methods proposed in this thesis. Study 2 was selected to illustrate the added flexibility the proposed annotation methods bring to mvGST analysis. Study 2 also provides an example to demonstrate the proposed SMP algorithmic alternative for handling annotation ambiguities.

All data from the two example studies are simple completely randomized designs and have been obtained through the public repository Gene Expression Omnibus on the NCBI website. A useful interface to this repository has been provided for R users with the package GEOquery [31].

2.1 Example Study 1

Example Study 1 has GEO accession number GSE47776. An experimental summary and the model set-up used in this thesis are subsequently provided.

The corpus luteum (CL) is a temporary structure in the ovary that forms with each

menstrual cycle. Degradation of the CL leads to abortion, and understanding the activity of biological processes in the CL is of interest to researchers of reproductive performance.

A study at Colorado State University tested the hypothesis, "gene expression differs in corpus luteum (CL) collected from pregnant (P) and non-pregnant (NP) ewes [4]." The Affymetrix Bovine Genome Array platform was used.

To test this hypothesis, six microarray samples were taken at Day 12 for three pregnant ewes and three non-pregnant ewes. Six more microarray samples were taken on Day 14 for three different pregnant ewes and three different non-pregnant ewes. The different treatments can be summarized as three replicates of the following:

$P12$: Pregnant ewes at Day 12

$P14$: Pregnant ewes at Day 14

$NP12$: Not Pregnant ewes at Day 12

$NP14$: Not Pregnant ewes at Day 14

Researchers involved in this study used their own models to test their hypothesis. In this thesis, the following contrasts can be constructed to test for the effect of pregnancy at each day by comparing expression profiles with the following contrasts:

1. $H_0 : P_{12} - NP_{12} = 0$
2. $H_0 : P_{14} - NP_{14} = 0$

Despite the study involving female sheep, the Affymetrix Bovine Genome Array platform was used to gather measures of gene expression. This array is GO-annotated and is supported by mvGST annotation methods. mvGST annotation methods will be compared to Biomart and Affymetrix annotation gathering methods in Chapter 3.1. The results can be found in Chapter 4.2.

2.2 Example Study 2

Example Study 2 has GEO accession number GSE52888. Additional platform annotation was provided by researchers at the Wood Lab of the University of Florida and was accessed by specifying the accession number GPL14112 [5]. This specific annotation is used in Chapter 3.2 in building a list of GO gene sets, the elements of which are the probeIDs of the microarray platform used in the study.

In this study, hypothalamic mRNA expression profiling was obtained using array Agilent-019921, a sheep gene expression array platform. Chronically catheterized fetal sheep were subjected to the following treatments:

C: Control (no treatment)

T_1: Brachiocephalic Occlusion (BCO) Treatment (10 min., no other treatment)

T_2: Estradiol Plus Brachiocephalic Occlusion Treatment

T_3: Estradiol Treatment (250 ug/day x 5 days)

There are four replicates of each treatment resulting in sixteen samples total. The Agilent array has 15008 unique probeIDs. In this thesis, meaningful contrasts to investigate the differences due to treatment type will be tested in the following way:

1. $H_0 : T_1 - C = 0$
2. $H_0 : T_2 - C = 0$
3. $H_0 : T_3 - C = 0$

The Agilent-019921 microarray is an example of what in this thesis will be termed a novel microarray for a non-model organism because probeID-to-GO annotation is not available using the current mvGST annotation methods. It should be noted that the terms "novel" and "non-model" are used differently elsewhere and were encountered in researching GO annotation files [19].

mvGST fails to annotate this platform for the following two reasons:

1. The organism *Ovis aries* is not a supported parameter in mvGST

2. Mappings from probeIDs to a GO annotated gene naming convention are unavailable with current mvGST annotation gathering methods.

Both constraints have been addressed with methods described in Chapter 3.2. For brevity, GO annotated gene naming conventions will be referred to as GOgncs.

When probeID-to-GO mappings are unavailable, a GOgnc can be used as a translator. mvGST has implemented a strategy that first takes probeIDs and maps them to GO-annotated Entrez IDs and then maps the probeIDs to GO terms. This strategy for annotating probeIDs to GO is limited in two ways. First, only Entrez is considered as a GOgnc translator. Second the function gconvert (used by mvGST to convert probeIDs to Entrez) is limited to non-novel platforms currently supported by Ensembl. The Agilent-019921 microarray is not supported in Ensembl's Biomart.

If a GOgnc translator must be used to map to GO terms, there is a potential for translation ambiguities to occur. Many-to-one and one-to-many translation ambiguities that have the potential to bias mvGST analyses are discussed in *mvGST: Tools For Multivariate and Directional Gene Set Testing* [20]. The thesis proposes and includes solutions within mvGST to handle translation ambiguities, but acknowledges them as temporary and somewhat unsatisfying. This has motivated the development in Chapter 3.2.4 of an SMP algorithmic approach for handling translation ambiguities by optimally choosing among ambiguous annotations based on measures of annotation strength.

Chapter 3

Methods

The first consideration in building GO gene set lists for use in mvGST is to consider the existence of probeID-to-GO annotation. If such annotation is available, then there is no need for use of a GOgnc translator to provide probeID-to-GO mappings. Consequently when such annotation is available, ambiguous potentially biasing mappings are avoided. Example study 1 is an example where such is the case. When probeID-to-GO annotation is not available, a GOgnc with annotation to the probeIDs on the microarray can be used to build a meaningful list of GO gene sets. These GO gene sets can then be used in mvGST analysis.

3.1 Example Study 1

To begin mvGST analysis of Study 1, an expression set matrix (see Chapter 1.3.1 and Table 1.4) was uploaded into R using the `getGEO` function of the GEOquery package and the study's GEO accession number.

Next, a matrix of p-values is obtained through use of the R package limma (see Chapter 1.4.1). Table 1.3 shows the header for this matrix of p-values. Notice that the contrasts of interest in the matrix of p-values represent the same contrasts described in Chapter 2.1. This matrix of p-values will be used here in three different calls to mvGST's `profileTable` function, for demonstration purposes.

ProbeID-to-GO annotation is available for the bovine Affymetrix platform used in example Study 1. The appropriateness of using a bovine microarray in a study involving sheep lies outside the scope of this thesis.

Example Study 1 was chosen specifically for comparison purposes. The following three main sources of probeID-to-GO annotation were considered in building the appropriate GO

gene set lists for this study:

1. mvGST (Chapter 3.1.1).
2. Ensembl's Biomart (Chapter 3.1.2).
3. Affymetrix (Chapter 3.1.3)

These three sources provide enough information to construct GO gene set lists, the differences of which are discussed in Chapter 4.

3.1.1 mvGST Annotation

The most convenient of all methods considered in this thesis for constructing a list of GO gene sets is to call the annotation arguments in mvGST's `profileTable` function. The function was designed to build the list automatically before mvGST analysis. An appropriate example of `profileTable` arguments for example study 1 are detailed below:

```
profileTable(pvals.study1, sig.level = 0.05, gene.ID = "affy",
          organism = "btaurus", affy.chip="bovine.db", ontology = "BP",
          method = 2, minsize = 1, maxsize = Inf, mult.adj = "BY")
```

By specifying these annotation arguments, this call to `profileTable` will build a list of GO gene sets for *Bos taurus* biological processes. It should be noted that elements of this list are GO terms where each is populated with the row names of the p-value matrix. In this example, genes within each GO gene set are Affymetrix probeIDs.

3.1.2 Biomart Annotation

The second method for constructing a list of GO gene sets for Study 1 considers the use of the R package biomaRt to query the Ensembl database. The following was used to appropriately query the Ensembl database for Study 1.

```
ensembl_80_btaurus <- useMart(biomart="ENSEMBL_MART_ENSEMBL",
          host="may2015.archive.ensembl.org",
```

```
          path="/biomart/martservice",
          dataset="btaurus_gene_ensembl")

annotation <- getBM(attributes=c('affy_bovine', 'go_id', 'namespace_1003'),
          values=T, mart = ensembl_80_btaurus)
```

The resulting structure for a similar example of a Biomart query can be seen in Chapter 1.6.3. Here the latest version of the `btaurus_gene_ensembl` dataset was used. The `getBM` function allows for specific queries to this organism-specific dataset. To build a list of GO gene sets similar to the list built with mvGST annotation methods, three attributes of annotation for the Affymetrix Bovine array must be obtained. They include Affymetrix probeIDs, GO IDs, and the domains for GO ID. Each row in this resulting data frame is unique and may contain cells of missing data. Before constructing the list of GO gene sets, it is appropriate to subset the data to only include rows with all three pieces of annotation information and limiting the rows to only include probeID-to-GO annotations describing biological processes. This sub-setting will ensure GO gene sets have at least one gene (equivalent to the `minsize=1` argument in mvGST's `profileTable` function).

The hierarchical structure of GO necessitates obtaining the offspring GO terms for each GO term already in the list and including them as separate gene sets in the list. This procedure was done using code from mvGST's `profileTable` function. The mvGST package gathers GO-offspring information using the GO.db package.

Computation time to query Biomart takes anywhere from 2-5 minutes and depends on the user's internet connection speed. The construction of this particular list of gene sets for the bovine Affymetrix array took about 5 minutes. This computation expense is larger but comparable to mvGST's annotation methods.

Below are gene sets five and six of the list constructed using this method. These sets correspond to specific biological processes and contain five and one gene(s), respectively.

```
$`GO:0000019`
[1] "Bt.27218.1.A1_at" "Bt.24555.1.S1_at" "Bt.17276.1.A1_at" "Bt.7008.1.S1_at"
```

```
"Bt.894.1.S1_at"
$‘GO:0045950‘
[1] "Bt.894.1.S1_at"
```

3.1.3 Affymetrix Annotation

The third method one may employ to gather probeID-to-GO annotation is to go straight to the microarray manufacturer. Depending on the structure of the annotation file provided by the manufacturer, this method can prove to be tedious.

A .csv file was downloaded from Affymetrix's annotation web page and read into R. The file is rather large at approximately 50 Mb and contains probeID-to-GO annotation. There are two columns in this file which contain all the information needed to build the GO gene set list for biological processes (BP). The most difficult part in using this file was extracting each unique GO term among descriptions of the GO terms. To accomplish the task of identifying each GO term and extracting them from the file, a regular expression was used to search for the 7 digit sequence common in all GO ID terms. The R code used to accomplish this task can be found in the Appendix. After all GO terms were extracted and put in list form, the original file was then used again to populate each GO term with probeIDs annotated to each GO term.

The computation time for this process took over an hour, much longer than the previous two methods considered in example Study 1. The results, however, were encouraging and are discussed further in Chapter 4.

3.2 Example Study 2

A matrix of p-values was obtained using the methods described in Chapters 1.3.1, 1.4.1, 1.4.2, and the contrasts described in 2.2. It was mentioned in Chapter 3 that when probeID-to-GO annotation is not available, then annotation from a GOgnc translator must be used (see Chapter 2.2). The idea is to follow probeID-to-GOgnc-to-GO in order to obtain probeID-to-GO annotation. When using a GOgnc translator, annotation ambiguities are introduced and should be corrected prior to mvGST analysis. The goal of the methods

described in the subsequent sections is to provide an SMP-algorithm-optimized list of GO gene sets for *Ovis aries* biological processes. The genes within each GO set of the resulting list will be probeIDs from the Agilent-019921 Sheep Gene Expression Microarray.

For this example, the complete computation time to construct the final GO gene set list is approximately 15 minutes. This expense includes multiple queries to Biomart.

3.2.1 ProbeID-to-GOgnc Annotation

ProbeID-to-GOgnc annotation for novel arrays are often not found in annotation repositories. Such is the case with the *Ovis aries* Agilent-019921 microarray. No annotation for this array is found in Ensembl's Biomart. The gene naming conversion methods used in mvGST also fail because they also use information provided by Ensembl (see Chapter 2.2).

Two annotation sources were obtained for Agilent-019921– the manufacturer's annotations and those provided by Charles Evans Wood's research group [5]. GOgnc candidates will be considered and compared from both sources of annotation, but first a quick glance at the annotation available under each gene naming convention (gnc) can be found in Tables 3.1 and 3.2.

Table 3.1: GOgnc Candidates: Agilent Source

	Annotated	Uniquely Annotated	Duplicated Annotation
ProbeID	1	1	0
PrimaryAccession	0.910	0.810	0.110
RefSeqAccession	0.470	0.320	0.330
GenbankAccession	0.860	0.770	0.110
UniGeneID	0.770	0.680	0.120
EntrezGeneID	0.210	0.110	0.490
GeneSymbol	0.210	0.110	0.490
GeneName	0.210	0.110	0.490
EnsemblID	0	0	0
TIGRID	0.680	0.560	0.180
GO	0.020	0.010	0.130
Description	0.900	0.800	0.110
GenomicCoordinates	0	0	0
Cytoband	0	0	0

Table 3.2: GOgnc Candidates: Wood Research Lab Source

	Annotated	Uniquely Annotated	Duplicated Annotation
ID	1	1	0
GeneName	1	0.920	0.080
ORF	0.820	0.390	0.530
Approved Name	0.810	0.380	0.530
GB_ACC	0.860	0.860	0
SPOT_ID	0.010	0.010	0.100

These tables were created using the function `quickView` provided in the Appendix. The "ProbeID" and "ID" naming conventions found in the first rows of Tables 3.1 and 3.2 identically contain the names of the probeIDs associated with the microarray Agilent-01992. It can be seen in the tables that each probeID (ID) is unique. This is important when considering probeID-to-GOgnc annotation ambiguities. The uniqueness of probeIDs in this example indicate that only one-to-many type ambiguities can occur. It is also of value to notice that only 0.020 of the probeIDs have some sort of GO annotation. This fact illustrates what is meant by probeID-to-GO annotation not being available.

Tables 3.1 and 3.2 not only provide the names of all possible GOgnc translators one might consider, but also give a preliminary indication as to which would make the best candidates. The column "Annotated" can be interpreted as showing the proportion of probeIDs that have some sort of annotation for each row gnc. EntrezGeneID, for example, has annotation for .21 of the probeIDs on the microarray. EntrezGeneID has unique annotation for only .11 of the probeIDs. Also, there are $0.210 * 15,008 = 3,152$ probeID-to-EntrezGeneID annotations, of which .49 are duplicated at least once. The column "Duplicated Annotations" is an indication of the magnitude of translation ambiguities present. More specifically, when taking into consideration the fact that all the probeIDs are unique, these duplications indicate the magnitude of potentially biasing one-to-many translation ambiguities.

One might conclude that the best GOgnc translator candidates are "PrimaryAccession", "GenbankAccession", "Unigene", "TIGRID" from Agilent and "GeneName", "ORF" and "GB_ACC" from the Wood research group. However, such conclusions based on these

tabulations are premature. The performance of each of these naming conventions in mapping to probeIDs has been quantified but not their ability to map to GO. The gene naming convention that most successfully maps to both probeIDs and GO terms is the most appropriate GOgnc translator.

GOgnc Potential Performance

GOgnc-to-GO annotations are obtained using Ensembl's Biomart in a Biomart query similar to that described in Chapter 3.1.2. The R function `gncTranslate` (see Appendix) was created to merge two annotation bi-maps, one for the probeID-to-GOgnc bi-map as provided by either Agilent or the Wood research group, and one for the GOgnc-to-GO bi-map as provided by Ensembl's Biomart. The output of this function is useful because it provides the annotation matrices needed to build the list of GO gene sets and also performance metrics for the GOgnc translator after mapping to GO. GenbankAccession annotations given by Agilent provide the best example of a gnc that has a high "Annotated", "Uniquely Annotated", and a low "Duplicated Annotation" combination, yet, it does not map well to GO.

The following was executed in R and yields translation performance metrics for GenbankAccession after mapping to GO in Table 3.3.

```
agilent.gnc.GB_ACC <- gncTranslate(probe.embl,embl.go,"embl",agilent$ProbeID)
agilent.gnc.GB_ACC[[3]]
```

Table 3.3: Translation Performance Metrics for GB_ACC

	gncs	probe_ids	annotations	go_ids	chipCoveragePotential
1	139	214	214	1097	0.01425906

The metric "chipCoveragePotential" in Table 3.3 is the number of microarray probeIDs with annotation to at least one GO term over the total number of unique probeIDs. The probeIDs with GO annotation have the "potential" to be used in mvGST analysis. After the

SMP procedure for Study 2 (see Section 3.2.4), the final "chipCoverage" metric is reported in Chapter 4.2.

It can be seen from Table 3.3 that there are only 139 unique GenBankAccession identifiers that map to probeIDs and GO terms of biological processes (BP). These 139 GenBankAccession identifiers map to 214 probeIDs and 1,097 different GO terms of BP. The matrix of p-values has a p-value associated with each probeID which means only 214 p-values will be used in subsequent mvGST analysis. This is only .0141 of the 15,008 probeIDs represented on this microarray chip. The best candidate for use in construction of the GO list will have the most GO terms and the highest chipCoveragePotential. The term potential is important because ambiguities have not yet been accounted for. After using the `smp` function that will be described in Chapter 3.2.4 the final chipCoverage will be reported in Chapter 4 and will always be less than the chipCoverage Potential.

By comparing gncs, probe_ids and annotations, one can get an idea of the magnitude of annotation ambiguity. As illustrated in Table 3.3 there are 214 probeID-to-GOgnc annotations and 214 probe_ids. One can conclude from this that each probeID-to-GOgnc annotation is unique because each probeID is unique. This is why both columns report the same number of 214. Notice that there are fewer GOgncs than probeIDs illustrating that there are many-to-one (many probeIDs to one GOgnc) annotation ambiguities only. In other words, there are no one-to-many ambiguities.

Table 3.4 gives the performance metrics for different GOgnc translators after mapping to GO BPs for each of the two sources of annotation information. The results are surprising in that the GOgncs that initially looked like good candidates from output provided by the `quickView` function, see Tables 3.1 and 3.2, turned out to map poorly to GO (i.e. had low chipCoveragePotential and low numbers of unique GO gene sets).

The structure of the Wood annotation file consists of rows that contain microarray probeIDs and columns that specify different gncs. The annotation provided by the Wood Research Group was difficult to work with because multiple GOgncs were found under a single column. The term ORF (a column in the file) in fact refers to Open Reading Frame

Table 3.4: Translation Performance Metrics for Candidate GOgncs

	GOgncs	probeIDs	Annotations	GO	chipCoveragePotential
Agilent GB_ACC	139	214	214	1097	0.01425906
Agilent Unigene	2661	3025	3025	5050	0.2015592
Agilent EntrezID	1627	2217	2217	4552	0.1477212
Agilent RefSeq	3030	3763	3763	5348	0.2507329
Wood GenBank	0	0	0	0	0
Wood ORF	5121	7298	7298	6698	0.486274

which is not even a gnc let alone a GOgnc. After exploring the "ORFgnc" in the Wood annotation file, the majority of entries were comprised of GenBankAccession numbers and HGNC symbols (HUGO Gene Nomenclature Committee Symbols [32]. It can be seen in Table 3.4 that the Wood GenBankAccession did not provide any annotations and many of these same GenBankAccession numbers are found in ORF. The reasonable step then was to consider only querying Biomart for HGNC Symbols found within the ORF column. This proved to be the best performing GOgnc with 6,698 unique GO terms, 5,121 unique GOgnc identifiers, and 7,298 unique probeIDs. The chipCoveragePotential was also by far the highest at 0.486274. These probeID-to-HGNC-to-GO annotations were selected to build GO gene sets.

3.2.2 Build GO Gene Sets

Here the function `groupBuilder` (see Appendix) was used to group probeID-to-HGNC annotations by GO terms. Using the results from Chapter 3.2.1, the call to `groupBuilder` took approximately 4-5 minutes. Some smaller elements of the resulting list are shown below.

```
$‘GO:0001869‘
           gnc     probe_id
3          A2M A_70_P017696
37941 SERPING1 A_70_P058611
$‘GO:0006103‘
```

```
         gnc     probe_id
9      AADAT A_70_P005451
42002 STAT5B A_70_P022711
42059 STAT5B A_70_P017016
42884    TAT A_70_P037681
```

The second GO term in the output ('GO:0006103') illustrates two ambiguous annotations, namely:

```
42002 STAT5B A_70_P022711
42059 STAT5B A_70_P017016
```

STAT5B is a HGNC symbol identifier that is annotated to two different probeIDs (A_70_P022711 and A_70_P017016). These annotation ambiguities within each GO term will be handled using the proposed SMP procedure (see Chapter 3.2.4).

3.2.3 Sequence Alignments as Measures of Annotation Strength

Prior to selecting the optimal annotations within each GO group, a measure of strength for each annotation must be calculated. It was proposed to use sequence alignment similarity scores as a way of measuring the strength or quality of each annotation. The `pairwiseAlignment` function in the Biostrings R package was used to conduct local-global sequence alignments. When considering what alignment type to perform, look at the Biostrings package help file. The local-global sequence alignment seemed most appropriate in this example as probeID sequences from this array are much smaller (60-mer) than the GO-gnc sequences.

Sequences were gathered in a Biomart query for each HGNC Symbol and each probeID found in the GO list produced in Chapter 3.2.2. The 60-mer oligonucleotides sequences for the probeIDs are all uniquely mapped in a 1-1 relationship. The HGNC symbols however were not. The cDNA transcripts associated with each HGNC Symbol are Ensembl transcripts. Some HGNC Symbols have multiple Ensembl cDNA transcripts. This observation

was noticed when the number of unique sequences (5718) and unique HGNC Symbols (5121) resulted in unequal numbers. It should be noted there are more transcript sequences than HGNC Symbols and also each HGNC Symbol is unique. This means there are only many sequences-to-one HGNC ambiguity type, as in the example in Chapter 3.2.2. This fact will be important when considering the proposed solution.

3.2.4 SMP

The SMP algorithm discussed in Chapter 1.7 will be applied to annotations in each GO term. Preferences are determined by sequence alignment scores. The procedure returns the maximum number of optimally "stable" 1-1 annotations within each GO term.

The solution chosen to handle sequence-gnc ambiguities was first to consider each annotation in the GO-grouped list from Chapter 3.2.2. For each HGNC in the list with multiple sequences, the sequence that maximized the alignment score when aligned to the probeID sequence to which it was annotated was chosen. This procedure seemed the natural solution for handling these sequence-gnc ambiguities in context of what is desired. This endeavor was initially very challenging and computationally expensive, inspiring the `annotator` function (in Appendix) to limit the amount of string matches made by the `smp` function. The changes in the manner of string comparison by the `annotator` function drastically reduced the computation time when calling the `smp` function.

There are multiple procedures embedded in the `smp` function and will not be discussed in detail here (see R code in the Appendix). As an overview, the function takes as arguments results from Chapter 3.2.3. The first procedure executed by `smp` is to obtain the maximum sequence alignment score for each annotation in each GO group. Then each probeID-to-GOgnc annotation (in this example probeID-to-HGNC Symbols) in each GO group is ordered according to GOgnc and then by alignment score. This ordering is important when building the matrix of preferences for each GOgnc (see Chapter 1.7). The same ordering methodology is done for the probeIDs to rank their preferences of GOgncs. Finally, for each GO group the `daa` function (see Chapter 1.7) is called and returns optimal probeID-to-GOgnc pairings. The result is the desired list of GO grouped probeIDs.

Similar to methods described in Chapter 3.1.2, the hierarchical structure of GO is accounted for using code from mvGST. This step is the last so the resulting list can then be used in mvGST analysis. Displayed below are some of the smaller elements of the resulting list.

```
$‘GO:0000002‘
 [1] "A_70_P062131" "A_70_P026981" "A_70_P071411" "A_70_P061636"
 [5] "A_70_P062171" "A_70_P022551" "A_70_P045001" "A_70_P025356"
 [9] "A_70_P020636" "A_70_P049666" "A_70_P049667" "A_70_P002736"
$‘GO:0000012‘
[1] "A_70_P063286" "A_70_P052106" "A_70_P039931" "A_70_P061911"
$‘GO:0000019‘
[1] "A_70_P010796" "A_70_P028691" "A_70_P024591"
```

This output contains three GO terms (GO:0000002, GO:0000012, GO:0000019) each of which correspond to a biological process. The elements within each GO term are the probeIDs corresponding to the Agilent-019921 microarray. The GO terms in the list could have easily been populated with HGNC symbols instead of microarray probeIDs. For greater interpretability, researchers may prefer GO gene set lists populated by gncs rather than probeIDs.

3.3 Custom R Annotation Packages

Creating custom organism-specific annotation packages within R was investigated as an annotation-gathering option. The idea was to create annotation packages for organisms currently not supported by mvGST and add them prior to mvGST analysis. This method was abandoned after creating annotation packages for Study 1 and Study 2.

Two annotation packages were created, one a *Bos taurus* package for Study 1 and another *Ovis aries* package for Study 2. These annotation packages were built using the

`makeOrgPackageFromNCBI` function from the AnnotationForge package [26]. Both annotation packages contained information for multiple gncs and transcript sequences, but as compared to Biomart the selection was limited.

The custom-built *Bos taurus* package did not contain Affymetrix-to-GO annotation unlike the annotation package for *Bos taurus* used by mvGST annotation methods. With Affymetrix-to-GO annotation being available through both Biomart and Affymetrix's own annotation file, the failure of the custom annotation package approach to provide direct GO annotation would have unnecessarily required a GOgnc translator. Translating unnecessarily would have introduced avoidable translation ambiguities. This result was taken as evidence that this method of annotation was not as complete as others considered in this thesis.

The custom *Ovis aries* package likewise did not contain probeID-to-GO annotation, but this time Biomart also failed to produce direct GO annotation. The main strike against the custom *Ovis aries* package was that it contained a paltry selection of GOgncs compared to Biomart. It became evident that a custom *Ovis aries* package would never be preferred to using Biomart as a method of implementing GOgnc translators.

As a final note, both packages took over three hours to compile. This expense can be compared to a Biomart query that ranges from 3-5 minutes.

Chapter 4

Discussion

For Study 1, the three GO gene set lists constructed in Chapter 3.1 are compared. The results of mvGST analysis using each of the three lists are also compared and discussed.

The performance metrics for the *Ovis aries* GO gene set list for Study 2 constructed in Chapter 3.2 are reported. mvGST results for Study 2 are also shown and discussed.

To conclude, a brief discussion about possible extensions to this work is presented.

4.1 Study 1 Discussion

Study 1 allowed for the comparison of three different GO grouped *Bos taurus* gene set lists. Somewhat surprisingly, the GO gene set lists vary widely depending on what method is used to build them. Two metrics were considered in assessing which list would be best to use in mvGST analysis– 1) the number of unique GO identifiers in each list and 2) the number of unique probeIDs in each list relative to the number of probeIDs on the microarray (chipCoverage).

The justification for choosing these two metrics is that no annotations were created from methods discussed in this thesis and so the probeID-to-GO annotations were simply gathered from different sources. Assessing the quality of annotation lies outside the scope of this thesis. Therefore, the annotation-gathering method that yields the highest quantity of annotation is considered the most appropriate. A brief note on annotation quality is mentioned in Chapter 4.3

Figure 4.1 illustrates a visual comparison for the three different GO gene set lists built in Chapter 3. The best-performing list in this figure would be the list closest to the upper right corner. What can be seen in the plot is that the annotation method provided by mvGST is the lowest performing method.

It is unclear which method is preferred when comparing the annotation provided by Biomart and Affymetrix. The list constructed using annotation from Affymetrix provides greater chipCoverage whereas the list constructed using annotation from Biomart has more unique GO identifiers.

4.1.1 Compare GO Grouped Lists

The list performance plot in Figure 4.1 should be used to select the most appropriate GO gene set list. Additionally, by comparing each of the three lists to each other, conclusions about annotation concordance among the lists can be reached.

To compare lists to each other, four different list grouping comparisons were made. They are: Biomart/Affy, Affy/mvGST, Biomart/mvGST, and Biomart/Affy/mvGST, see Figure 4.2. Three criteria were used in the comparisons– 1) GO IDs, 2) probeIDs, and 3) Gene Sets. The proportion concordant for GO ID terms is the number of unique GO ID terms that match in all lists of a given comparison over the number of unique GO ID terms found in any one of the lists of a given comparison. The proportion concordant for the probeID criterion is similar– it's the number of probeIDs common to all lists in a list grouping comparison over the number of probeIDs found in any one of the lists in a list grouping comparison. The Gene Sets criterion considers both GO IDs and their probeID elements. The proportion concordant for Gene Sets is the number of GO IDs with exactly the same probeID elements in all the lists of a given comparison over the number of GO IDs found in any one of the lists in the comparison.

Figure 4.2 illustrates that the GO gene set lists constructed using annotation from Biomart and Affymetrix are the two most similar lists. Of the total unique GO IDs and probeIDs among these two lists, approximately .78 and .74 of them are in both lists, respectively. Other list comparisons have lower values for the GO ID and probeID criteria. When considering the comparison groupings, this decrease is attributable to the mvGST method of gathering annotation.

The fourth grouping (Biomart/Affymetrix/mvGST) is an interesting comparison because it indicates the proportion of concordance among all the lists. The proportion of

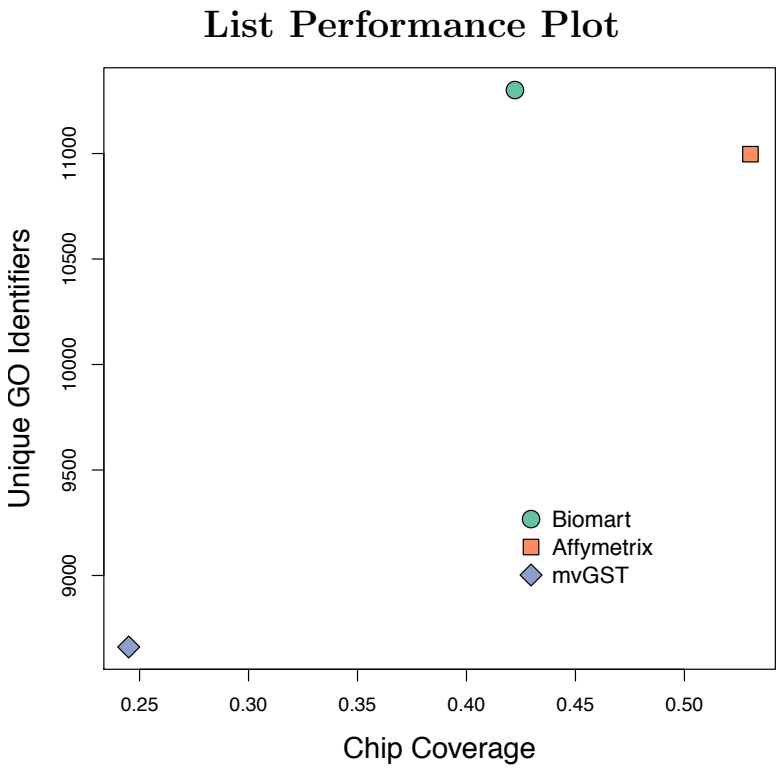

Fig. 4.1: List performance is based on two metrics: 1) the total unique GO Identifiers included in the list (vertical-axis) and 2) how many unique probeIDs are included in at least one GO gene set (horizontal-axis). The best performing list will have the highest values for both.

concordance for GO IDs and probeIDs seems low, just less than .75, but most surprising might be the proportion of gene sets common to all three lists. This value stands at less than two percent. This result indicates that less than two percent of the GO gene sets among the lists are evaluated equally when mvGST analysis is performed.

4.1.2 Compare mvGST Analysis Results

The proportion of GO gene sets for each list identified as significant (or not) by mvGST analysis can be seen in Figures 4.3 and 4.4 for Day 12 and Day 14, respectively. The results from mvGST analysis were similar for each list in the proportion it identified as either not significant, more active, or less active in Day 12; however, the list built by mvGST annotation methods showed a slightly greater proportion of the GO gene sets identified as more active compared to the other two lists. Consequently, the added proportion identified as more active corresponds to a lesser proportion of GO gene sets identified as less active as compared to the other two lists. These observations further demonstrate that the similarity between the Affymetrix and Biomart created lists, shown in Chapter 4.1.1, can be seen after mvGST analysis.

Figures 4.3 and 4.4 also demonstrate that the dissimilarity of the gene sets among the lists doesn't overly bias the proportion of gene sets identified as either significantly more active, less active, or not differentially active as the proportions are very similar regardless of which list was used in the analysis.

4.2 Study 2 Discussion

Study 2 provided an example of a novel microarray whose probeIDs could not be directly mapped to GO. As a solution, a GO-annotated gnc was selected as a translator of sorts to map probeIDs to GO terms. Different GO-annotated gncs performed with varying results as translators for the microarray. It has been mentioned that the ORF column in the annotation file provided by the Wood lab is not a gnc and is mostly comprised on HGNC symbols (see Chapter 3.2.1). The HGNC symbol gnc was selected as the best-performing-gnc translator because it yielded the highest performance metrics (see Table 3.4).

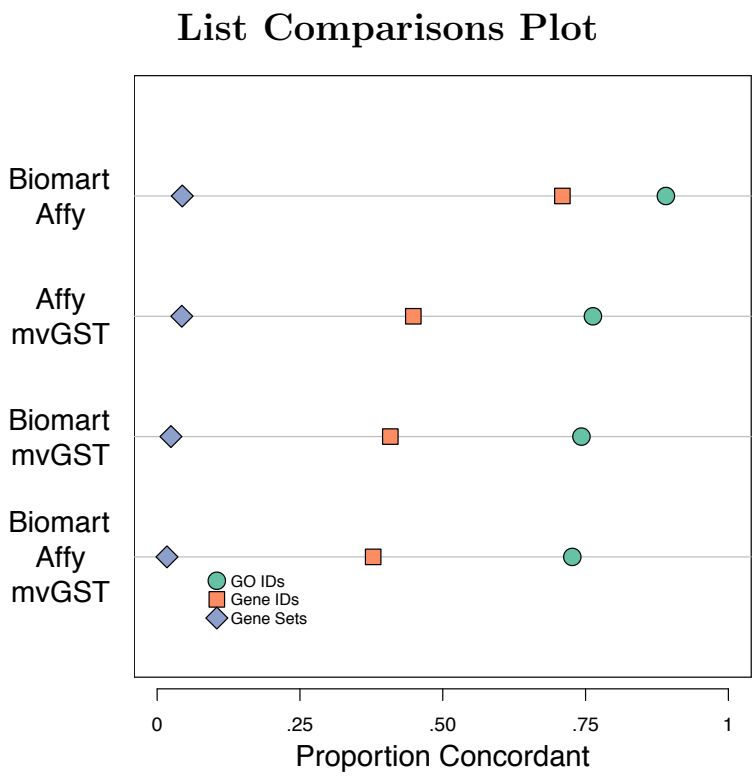

Fig. 4.2: Proportion concordant signifies the proportion of either GO IDs, Gene IDs, or Gene Sets that are found in all lists within the different list comparisons (found at vertical axis). GO IDs indicate the unique names of GO terms. Gene IDs are the unique gene identifiers. Gene sets consider both GO identifiers and their gene identifier elements.

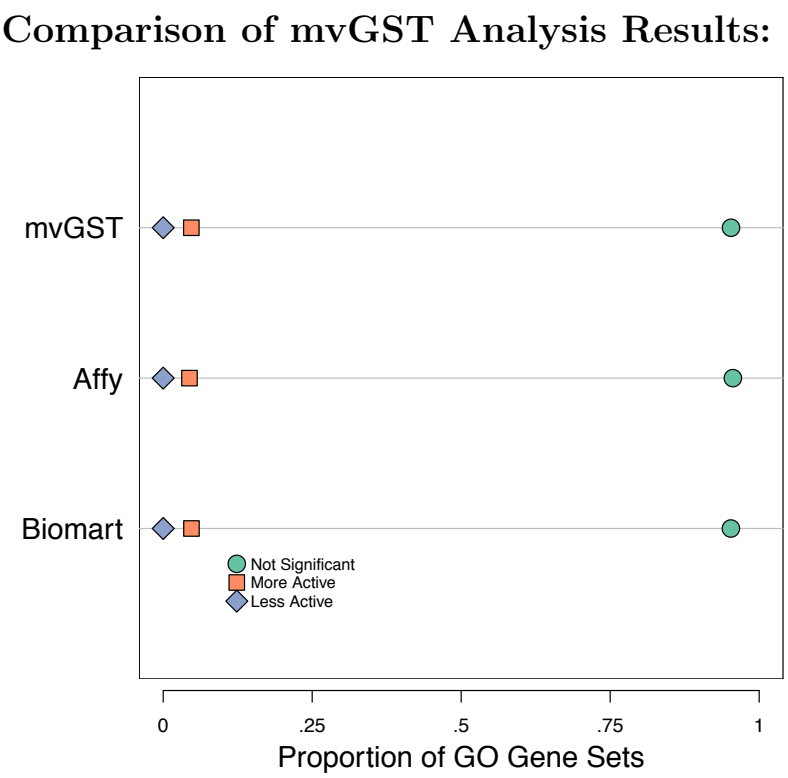

Fig. 4.3: Illustrates the proportion of GO Gene Sets that are either significantly more active, less active, or not differentially active in each list after mvGST analysis. Further explanation of the contrast of interest tested here, $H_0 : P_{12} - NP_{12} = 0$, can be found in 2.1.

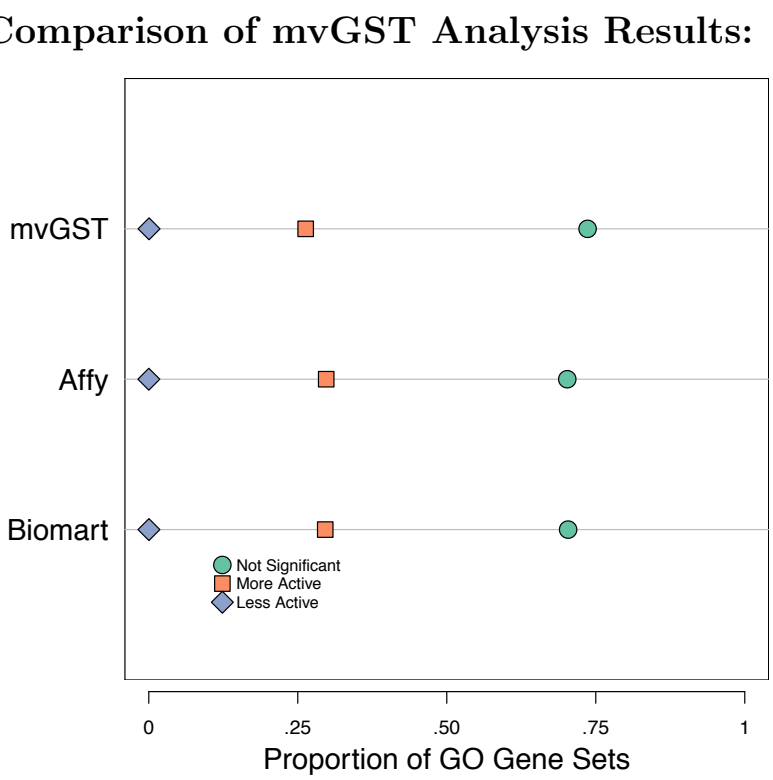

Fig. 4.4: Illustrates the proportion of GO Gene Sets that are either significantly more active, less active, or not differentially active in each list after mvGST analysis. Further explanation of the contrast of interest tested here, $H_0 : P_{14} - NP_{14} = 0$, can be found in Chapter 2.1.

Sequences were then gathered for each probeID and each HGNC symbol identifier. Working within the probeID-to-HGNC symbol annotation framework, sequence alignments were performed to get measures of strength for each annotation. The maximum measure was selected for HGNC symbols with multiple sequences. The SMP method for selecting the most appropriate 1-1 mapping between probeIDs and HGNC symbols was then employed for each GO term containing at least one HGNC symbol. Instead of populating GO terms with HGNC symbols, probeIDs were used to populate the GO gene set list. Parent GO terms of GO terms of this list were then obtained and populated with their GO term offspring probeIDs accounting for the hierarchical structure of the gene ontology.

4.2.1 GO Gene Set List: *Ovis aries* Microarray

The *Ovis aries* Agilent-019921 microarray was used in Study 2. A summary of the constructed GO gene set list can be found in Table 4.1.

Table 4.1 shows that all GO identifiers in the list are unique. This was expected and only mentioned here as a check. The chipCoverage metric is truly interesting. The chipCoverage metric is defined as the number of probeIDs used in mvGST analysis over the total number of probeIDs on the microarray. Notice that the chipCoverage for this list is 0.388. This value is approximately ten percent lower than the chipCoveragePotential reported in table 3.4 (the chipCoveragePotential in Wood ORF). This loss in chipCoverage is attributable to the SMP procedure described in Chapter 3.2. It is important to note that the unique number of probeIDs included in the final analysis is 5,823, a number between the reported number of unique GOgncs (5,121) and probeIDs (7,298) prior to the SMP procedure. If the SMP method were done on the annotations prior to grouping them by GO, the result would have been a total of 5,121 unique probeIDs used in mvGST analysis due to the fact that the SMP method always returns a 1-1 mapping. Though somewhat computationally expensive, doing the SMP procedure within each GO term allowed 702 probeIDs to be included in the final analysis that otherwise would not have been.

Table 4.1: Agilent-019921 GO Gene Set List Summary

Unique probeIDs	GO IDs	GO IDs	chipCoverage
5,823	10,082	10,082	0.388

4.2.2 mvGST Results

The results of mvGST analysis for Study 2 are stand-alone results and are reported in Table 4.2 for completeness. An example of interpretation from Table 4.2 can be expressed– there are 24 biological processes differentially more active (indicated by a 1) in the estradiol-plus-brachiocephalic-occlusion (EBO) treatment group with no significant activity difference (indicated by a 0) in the brachiocephalic-occlusion (BO) and estradiol (E) groups, when compared to the control group. The other four rows can give similar interpretations.

For further methods to identify which gene sets are included in each profile, see *mvGST: Tools For Multivariate and Directional Gene Set Testing* [20].

Table 4.2: mvGST Analysis Results for Study 2

BO-C	EBO-C	E-C	BP
0	0	0	9,745
0	0	1	271
0	1	1	42
0	1	0	24

Without the methods and tools developed as part of this thesis, the *Ovis aries* results given in Table 4.2 could not have been obtained using mvGST because Study 2 was conducted with a less traditional platform in a non-model organism.

4.3 Potential Extensions

In the process of developing the methods described in this thesis, a few possible extensions were discovered but not fully explored.

In constructing the GO gene set list for Study 2, probeID-to-GO-gnc (HGNC symbols) annotation was considered. There is no compelling reason that multiple GO-gnc translators

could not be used. Doing so would likely increase both final chipCoverage and the number of GO terms used in mvGST analysis.

Using multiple GO-gnc translators would use more completely the annotation available. Where one probeID failed to map to a specific GO-gnc, there may be a mapping of that same probeID to a different GO-gnc. The SMP procedure would ensure that multiple p-values associated with the same probeID would not be included multiple times in any given GO gene set. Multiple queries to Biomart may have to be done to accomplish this task. Nevertheless, resulting lists constructed using multiple GO-gncs would be more complete.

When considering Agilent RefSeq as a GO-gnc translator in Chapter 3.2, two queries were made to Biomart because it was unclear which gnc in Biomart was appropriate. The code used to query Biomart for RefSeq can be used as an example of using multiple GOgncs as translators.

Another possible extension is to consider how measures of annotation appropriateness are to be calculated. These measures provide mapping preferences used in the SMP procedure for each probeID and each GO gene identifier. They ultimately determine how ambiguities are handled. In this thesis, the `pairwiseAlignment` function (from the Biostrings package [33]) returned sequence similarity scores for type *local-global* alignments as measures of annotation appropriateness. More sophisticated methods for assessing annotation strength may be employed and used by the SMP procedure presented in this thesis.

References

[1] Maciejewski, H., "Gene set analysis methods: statistical models and methodological differences," *Bioinformatics*, 2013.

[2] Lodish, H., Berk, A., Zipursky, S., and et al., *Molecular Cell Biology*, New York: W. H. Freeman, http://www.ncbi.nlm.nih.gov/books/NBK21640/, 4th ed., 2000.

[3] Robinson, P. N. and Baur, S., *Introduction to Bio-Ontologies*, CRC Press, 2011.

[4] Romero, J. J., Antoniazzi, A. Q., Smirnova, N. P., Webb, B. T., Yu, F., Davis, J. S., and Hansen, T. R., "Pregnancy-associated genes contribute to antiluteolytic mechanisms in ovine corpus luteum," *Physiological Genomics*, Vol. 45, No. 22, 11 2013, pp. 1095–1108.

[5] Wood, C., Rabaglino, M., Richards, E., Denslow, N., and et al., "Transcriptomics of the fetal hypothalamic response to brachiocephalic occlusion and estradiol treatment," *Physiol Genomics*, Jul 15 2014, pp. 523–32.

[6] Gundogdu, O. and Elmi, A., "Genome Resource Facility," http://grf.lshtm.ac.uk/microarrayoverview.htm, January 2015.

[7] Wu, Z., "A Review of Statistical Methods for Preprocessing Oligonucleotide Microarrays," *Statistical methods in medical research*, Vol. 186, 2009, pp. 533–541.

[8] Irizarry, R. A., Hobbs, B., Collin, F., Beazer-Barclay, Y. D., Antonellis, K. J., Scherf, U., and Speed, T. P., "Exploration, normalization, and summaries of high density oligonucleotide array probe level data," *Biostatistics*, Vol. 4, No. 2, 2003, pp. 249–264.

[9] Owzar, K., Barry, W. T., Jung, S.-H., Sohn, I., and George, S. L., "Statistical Challenges in Preprocessing in Microarray Experiments in Cancer," *Clinical Cancer Research*, Vol. 14, 2008.

[10] Vardhanabhuti, S., Blakemore, S., Clark, S., Ghosh, S., Stephens, R., and Rajagopalan, D., "A comparison of statistical tests for detecting differential expression using Affymetrix oligonucleotide microarrays," *Omics*, Vol. 10, No. 4, 2006, pp. 555–566.

[11] Fischer, E., Friedman, M. A., and Markey, M. K., "Empirical comparison of tests for differential expression on time-series microarray experiments," *Genomics*, Vol. 89, No. 4, 2007, pp. 460–470.

[12] Gentleman, R., Carey, V. J., Huber, W., Irizarry, R. A., and Dudoit, S., editors, *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, Statistics for Biology and Health, Springer New York, 2005.

[13] Tusher, V. G., Tibshirani, R., and Chu, G., "Significance analysis of microarrays applied to the ionizing radiation response," *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 98, No. 9, 2001, pp. 5116–5121.

[14] Smyth, G., "limma: Linear Models for Microarray Data," *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, edited by R. Gentleman, V. J. Carey, W. Huber, R. A. Irizarry, and S. Dudoit, Statistics for Biology and Health, Springer New York, 2005, pp. 397–420.

[15] Team, R. D. C., *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2008, ISBN 3-900051-07-0.

[16] Gentleman, R. C., Carey, V. J., Bates, D. M., and others, "Bioconductor: Open software development for computational biology and bioinformatics," *Genome Biology*, Vol. 5, 2004, pp. R80.

[17] Law, C. W., Chen, Y., Shi, W., and Smyth, G. K., "voom: precision weights unlock linear model analysis tools for RNA-seq read counts," *Genome Biology*, Vol. 15, No. 2, 2014.

[18] Richards, A. J., Muller, B., Shotwell, M., Cowart, L. A., Rohrer, B., and Xinghua, L., "Assessing the functional coherence of gene sets with metrics based on the Gene Ontology graph," *Bioinformatics*, Vol. 26, No. 10, 2010.

[19] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., and Sherlock, G., "Gene Ontology: tool for the unification of biology," *Nat Genet*, Vol. 25, No. 1, 05 2000, pp. 25–29.

[20] Mecham, D. S., *mvGST: Tools For Multivariate and Directional Gene Set Testing*, Master's thesis, Utah State University, http://digitalcommons.usu.edu/gradreports/382, 2014.

[21] "Gene Expression Omnibus," http://www.ncbi.nlm.nih.gov/geo/, Accessed: 2015-05-20.

[22] "Agilent," http://www.chem.agilent.com/cag/bsp/gene_lists.asp, Accessed: 2015-05-20.

[23] "Affymetrix," http://www.affymetrix.com/support/technical/annotationfilesmain.affx, May, Accessed: 2015-5-20.

[24] Cunningham, F., Amode, M. R., and et al., "Ensembl 2015," *Nucleic Acids Research*, Vol. 43, No. D1, May 2015, pp. 662–669.

[25] Durinck, S., Spellman, T., Birney, E., and Huber, W., "Mapping identifiers for the integration of genomic datasets with the R Bioconductor package biomaRt," *Nature Protocols*, Vol. 4, 2009, pp. 1184–1191.

[26] Carlson, M. and Pages, H., *AnnotationForge: Code for Building Annotation Database Packages*, R Foundation, R package version 1.8.1.

[27] "OrganismDbi in Bioconductor," May 2015.

[28] Gusfield, D. and Irving, R., "The Stable Marriage Problem: Structure and Algorithms," *MIT Press*, 1989, pp. 54.

[29] Gale, D. and Shapley, S., "College Admissions and the Stability of Marriage," *American Mathematical Monthly*, Vol. 69, 1962, pp. 9–14.

[30] Klein, T., *matchingMarkets: An R package for the analysis of stable matching*, 2014, R package version 0.1-2.

[31] Davis, S. and Meltzer, P., "GEOquery: a bridge between the Gene Expression Omnibus (GEO) and Bioconductor," *Bioinformatics*, Vol. 14, 2007, pp. 1846–1847.

[32] "HGNC," http://www.genenames.org/about/faq/#whatisthehgnc, Accessed: 2015-05-20.

[33] Pages, H., Aboyoun, P., Gentleman, R., and DebRoy, S., "Biostrings: String objects representing biological sequences, and matching algorithms," R package version 2.34.0.

Appendix

```
library(GEOquery)
library(limma)
library(mvGST)

################################
#Obtain Data for Example Study 1
################################

#Get the expression data#
EsetGSE47776 <- getGEO("GSE47776",GSEMatrix=TRUE)
#It reads in as a list of length one. Point to the ExpressionSet and save as an
    object
#so that it can be used in the global environment.
EsetGSE47776 <- EsetGSE47776$GSE47776_series_matrix.txt.gz

#Some functions used to explore an expression set object
#class(EsetGSE47776)
# featureNames(EsetGSE47776)[1:5]
# fvarLabels(EsetGSE47776)
#sampleNames(EsetGSE47776)
#varLabels(EsetGSE47776)

#####################################################################
#Limma\eBayes to test DE among genes across contrasts of interest
#Results in a matrix of p-values
#####################################################################

#Extract matrix of expression values
mat <- exprs(EsetGSE47776)
#Name columns appropriately where each column identifys a particular sample
```

```
colnames(mat) <- c("Day12.P","Day14.P","Day12.NP","Day12.NP","Day12.NP","Day14.P",
    "Day14.NP","Day12.P","Day12.P","Day14.NP","Day14.P","Day14.NP")
#Define treatments, fit linear model
Treatment <- as.factor(c("Day12.P","Day14.P","Day12.NP","Day12.NP","Day12.NP","
    Day14.P","Day14.NP","Day12.P","Day12.P","Day14.NP","Day14.P","Day14.NP"))
#Define design matrix
design <- model.matrix(~0+Treatment)
colnames(design) <- c('Day12.NP','Day12.P','Day14.NP','Day14.P')
#Fit linear model
fit <- lmFit(mat, design)

#Test 1st contrast
contrast1 <- makeContrasts(Day12.P-Day12.NP, levels=design)
fit1 <- contrasts.fit(fit, contrast1)
final.fit1 <- eBayes(fit1)
top1 <- topTableF(final.fit1, adjust.method="BH", n=nrow(mat), sort.by="none")

#Test 2nd contrast
contrast2 <- makeContrasts(Day14.P-Day14.NP, levels=design)
fit2 <- contrasts.fit(fit, contrast2)
final.fit2 <- eBayes(fit2)
top2 <- topTableF(final.fit2, adjust.method="BH", n=nrow(mat),sort.by="none")

#Prepare matrix of p-values
pvals.study1 <- cbind(top1$P.Value, top2$P.Value)
colnames(pvals.study1) <- c("Day12.P-Day12.NP","Day14.P-Day14.NP")
rownames(pvals.study1)<- rownames(top1)

##############################################################################
#Use mvGST annotation methods to build GO gene set list for example study 1
#Analyse this list with mvGST statistical methods
##############################################################################
```

```
#Slightly modify profileTable to not only evaluate the GO gene set list it creates
#but also return it and make it available outside of the function.
profileTable <- function (pvals, gene.names = NULL, contrasts = NULL, list.groups
    = NULL,
         sig.level = 0.05, gene.ID, organism, affy.chip, ontology = "BP",
         method = 2, minsize = 1, maxsize = Inf, mult.adj = "BY")
{
  if (is.null(contrasts)) {
    contrasts <- colnames(pvals)
  }
  if (is.null(gene.names)) {
    gene.names <- rownames(pvals)
  }
  if (!is.matrix(pvals)) {
    stop("pvals must be a matrix")
  }
  if (!is.element(mult.adj, c("BY", "SFL"))) {
    stop("mult.adj must be one of 'BY' or 'SFL'")
  }
  vars <- length(unlist(strsplit(contrasts[1], "[.]")))
  if (vars == 1) {
    contrasts <- paste(contrasts, ontology, sep = ".")
  }
  f.one <- 1 - .Machine$double.eps
  f.zero <- .Machine$double.eps
  pvals <- ifelse(pvals == 1, f.one, pvals)
  pvals <- ifelse(pvals == 0, f.zero, pvals)
  if (is.null(list.groups)) {
    if (any(gene.ID == c("affy", "genbank", "alias", "ensembl",
                         "entrez", "symbol", "genename", "unigene")) != TRUE) {
      old.names <- character()
```

```
new.names <- character()
for (i in seq.int(from = 0, to = floor(length(gene.names)/1000))) {
  low <- i * 1000 + 1
  high <- (i + 1) * 1000
  if (high > length(gene.names)) {
    high <- length(gene.names)
  }
  gene.names <- toupper(gene.names)
  converted1 <- gconvert(gene.names[low:high],
                      target = "ENTREZGENE_ACC", organism = organism)
  old.names <- c(old.names, as.character(converted1$alias))
  new.names <- c(new.names, as.character(converted1$target))
}
if (grep("^[[:digit:]]+$", str_trim(gene.names[1])) ==
     1) {
  all.numeric <- TRUE
}
new.genes <- cbind(old.names, new.names)
new.genes[, 2] <- gsub("ENTREZGENE_ACC:", "", new.genes[,2])
if (all.numeric) {
  new.genes[, 1] <- substr(new.genes[, 1], regexpr("[[:digit:]]+$",
                                          new.genes[, 1]), nchar(new.
                                              genes[, 1]))
}
duplicate <- rep(FALSE, length(new.genes[, 1]))
for (i in seq_along(new.genes[, 1])[-1]) {
  duplicate[i] <- ifelse(all(new.genes[i, ] ==
                             new.genes[i - 1, ]), TRUE, FALSE)
}
new.genes <- new.genes[!duplicate, ]
converted <- geneNameConvertRows(pvals, gene.names,
                              new.genes, method)
```

```
    pvals <- converted$p.mat
    gene.names <- converted$genes
    gene.ID <- "entrez"
  }
  list.groups1 <<- generateGeneSets(ontology = ontology,
                                species = organism, ID = gene.ID, affy.chip)
  offspring <- get("as.list", pos = "package:AnnotationDbi")(get(paste("GO",
                                                          ontology, "
                                                              OFFSPRING",
                                                              sep = "")))
  list.groups <- sapply(1:length(offspring), function(x) fillInList(list.groups1
      [[names(offspring[x])]],
                                                        names(offspring)[x
                                                            ], offspring,
                                                            list.groups1))
  names(list.groups) <- names(offspring)
  size <- sapply(list.groups, length)
  list.groups <<- list.groups[(size >= minsize) & (size <=
                                                 maxsize)]
}
pmat <- mvGSTObject(gene.names, contrasts, pvals, list.groups)
grouped.pmat <- separate(pmat, list.groups)
if (mult.adj == "SFL") {
  adjusted <- grouped.pmat
  adjusted$adjusted.group.pvals <- grouped.pmat$grouped.raw
  adjusted$adjusted.group.pvals[] <- NA
  ncgp <- ncol(grouped.pmat$grouped.raw)
  for (i in seq_len(ncgp)) {
    pvalues <- grouped.pmat$grouped.raw[, i]
    two.sided <- convertPvalues(pvalues, two.sided = FALSE)
    names(two.sided) <- grouped.pmat$group.names
    new.pvalues <- p.adjust.SFL(two.sided, sig.level = sig.level)
```

```
      relative <- ifelse(pvalues < 0.5, 1, -1)
      one.combined <- convertPvalues(new.pvalues, relative)
      adjusted$adjusted.group.pvals[, i] <- one.combined
    }
  }
  else if (mult.adj == "BY") {
    adjusted <- oneSideBYAdjust(grouped.pmat)
  }
  one.zero <- changeTO10(adjusted, sig.level = sig.level)
  almost.final <- finalResults(one.zero)
  near.final <- cut(almost.final)
  final <- mvSort(near.final)
  return(final)
}

#Generate mvGST analysis results and return the GO gene set list used in the
    analysis
results.mvGST <- profileTable(pvals.study1, sig.level = 0.05, gene.ID = "affy",
    organism = "btaurus",
                          affy.chip="bovine.db", ontology = "BP", method = 2,
                              minsize = 1,
                          maxsize = Inf, mult.adj = "BY")

#Save matrix of p-values to be used latter in mvGST analysis with different GO
    gene set lists
#Rename list.groups to identify list was built for Bos taurus with mvGST
    annotation methods
#Save the renamed list
save(pvals.study1, file="/Users/russell/Desktop/USU␣Student␣Materials/Research␣
    Summer␣2014/Thesis/R␣Code/pvals.study1")
profileTable.btaurus <- list.groups
```

```
save(profileTable.btaurus,file="/Users/russell/Desktop/USU␣Student␣Materials/
    Research␣Summer␣2014/Thesis/R␣Code/profileTable.btaurus")

load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/pvals.study1")
load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/profileTable.btaurus")

################################################################################
#Create GO gene set list using biomart annotation method for example study 1
#Test the list for DE among GO gene sets using mvGST
################################################################################
library(biomaRt)
library(dplyr)

#To access archived databases it's recommended to set the host to the desired
    archived Ensembl website
#A list of archived websites can be found at: http://www.ensembl.org/info/website/
    archives/index.html
#80 or may2015 is the most recent
ensembl_80 = useMart(biomart="ENSEMBL_MART_ENSEMBL", host="may2015.archive.ensembl
    .org",
                     path="/biomart/martservice")

#View list of datasets available in this biomart version
#listDatasets(ensembl_80)
#Check number of different species are supported in this biomart version
#dim(ensembl_80)

#Qeury Ensembl database for Bos taurus
ensembl_80_btaurus <- useMart(biomart="ENSEMBL_MART_ENSEMBL", host="may2015.
    archive.ensembl.org",
```

```
                        path="/biomart/martservice", dataset="btaurus_gene_
                            ensembl")

#Databases are large, check dimension first before exploring attributes
#dim(listAttributes(ensembl_80_btaurus))
#See what what attributes are available for Bos taurus
#This argument is where gnc options can be found
#listAttributes(ensembl_80_btaurus)[1:100,]

#Creates data frame of selected items where rows are different annotation mappings
#For example study 1, the microarray is Affymetrix Bovine. "namespace_1003" is GO
    domain
#Query takes less than a minuete.
annotation <- getBM(attributes=c('affy_bovine', 'go_id', 'namespace_1003'),
                  values=T, mart = ensembl_80_btaurus)

#The resulting data frame has missing values. Each column's missing values either
#are represented by "" or NA but never both. Missing values can be avoided with
    specific
#arguments passed to getBM, but subsetting must be done to include on BP GO so it'
    s easier
#to just subset out rows with missing values at the same time.

new.annot <- filter(select(annotation, affy_bovine, go_id, go_domain=namespace_
    1003),
                  affy_bovine != "", go_id != "", go_domain == "biological_process
                      ")

#Create GO gene set lists
listBuilder <- function (annotations) {
  x <- annotations$go_id
  y <- annotations$affy_bovine #specific to the probeIDs or gnc
```

```
  uniques <- unique(x)
  groupList <- setNames(vector("list", length(unique(x))), uniques)
  finish <- length(uniques)
  for (i in 1:finish)
  {
    t <- grep(uniques[i], x)
    groupList[[i]] <- y[t]
  }

  #The hierarchical structure of GO necessitates getting the offspring GO terms
      and including them in the list
  offspring <- get("as.list", pos = "package:AnnotationDbi")(get(paste("GO", "BP",
       "OFFSPRING", sep = "")))
  groupList1 <- sapply(1:length(offspring), function(x) fillInList(groupList[[
      names(offspring[x])]],
                                                                   names(offspring)[x],
                                                                       offspring,
                                                                       groupList))
  names(groupList1) <- names(offspring)
  #Only include gene sets with at least one gene
  final.list <- groupList1[sapply(groupList1, length)!=0]

  return(final.list)
}
#Takes about 4 minutes
ptm <- proc.time()
groupList <- listBuilder(new.annot)
proc.time() - ptm

#Check list for any duplicated probeIDs within each GO gene set
sum(unlist(lapply(groupList, function (x) sum(duplicated(x)))))
```

```
#Save list with name that identifys some of the list's attributes and also so you
#don't have to create the list until a new version of biomart is released
biomart.btaurus <- groupList
save(biomart.btaurus,file="/Users/russell/Desktop/USU␣Student␣Materials/Research␣
    Summer␣2014/Thesis/R␣Code/biomart.btaurus")

load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/biomart.btaurus")
results.biomart <- profileTable(pvals.study1, list.groups=biomart.btaurus, sig.
    level = 0.05, mult.adj = "BY")

##############################################################################
#Create GO gene set list using affy annotation method for example study 1
#Test the list for DE among GO gene sets using mvGST
##############################################################################

#This file was downloaded on May 26 2015 from the Affymetrix website
#found here: http://www.affymetrix.com/support/technical/annotationfilesmain.affx
annot <- read.csv(file="/Users/russell/Desktop/USU␣Student␣Materials/Research␣
    Summer␣2014/Data/Bovine/Bovine.na35.annot.csv",head=TRUE,sep=",", skip=19)

sub <- as.matrix(select(annot,
                    probe_id = Probe.Set.ID,
                    go_id_bp = Gene.Ontology.Biological.Process))

#Get go_id terms out of column gene.onotology.biological.process with regular
    expression
go_id <- list()
finish <- length(sub[,2])
for (i in 1:finish) {
  go_id[[i]] <- unlist(strsplit(as.character(sub[,2][i]), "[^0-9]+"))
}
```

```
#Build GO terms gene set list
#Takes over an hour
gene.set.list <- setNames(vector("list", sum(nchar(unique(unlist(go_id)))==7)),
                          unique(unlist(go_id))[nchar(unique(unlist(go_id)))==7])
finish <- length(gene.set.list)
for (i in 1:finish) {
  t <- grep(names(gene.set.list[i]), go_id)
  gene.set.list[[i]] <- sub[t,1]
}
names(gene.set.list) <- paste0("GO:", names(gene.set.list))

#The hierarchical structure of GO necessitates getting the offspring GO terms and
    including them in the list
offspring <- get("as.list", pos = "package:AnnotationDbi")(get(paste("GO", "BP", "
    OFFSPRING", sep = "")))
full.gene.set.list <- sapply(1:length(offspring), function(x) fillInList(gene.set.
    list[[names(offspring[x])]],
                                                                          names(offspring)
                                                                              [x],
                                                                              offspring,
                                                                              gene.set.list
                                                                              ))
names(full.gene.set.list) <- names(offspring)

#Only include gene sets with at least one gene
final.list <- full.gene.set.list[sapply(full.gene.set.list, length)!=0]

#Check list for any duplicated probeIDs within each GO gene set
sum(unlist(lapply(final.list, function (x) sum(duplicated(x)))))

#Save list with meaningful name
```

```
#Saving this list is important because it takes over an hour to create!!
affy.btaurus <- final.list
save(affy.btaurus,file="/Users/russell/Desktop/USU␣Student␣Materials/Research␣
    Summer␣2014/Thesis/R␣Code/affy.btaurus")

load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/affy.btaurus")
results.affy <- profileTable(pvals.study1, list.groups=affy.btaurus, sig.level =
    0.05, mult.adj = "BY")

library(GEOquery)
library(mvGST)
library(GO.db)
library(gProfileR)
library(biomaRt)
library(Biostrings)
library(microbenchmark)
library(matchingMarkets)
library(limma)
library(dplyr)

#select() is masked by the following packages: biomaRt, AnnotationDbi, MASS
#move select() from dplyr to gobal environment so it's always called first.
select <- dplyr:::select

##############################################################################

#Annotation for Agilent-019921 Microarray

#Agilent annotation file downloaded from website: https://earray.chem.agilent.com/
    earray/
#on Monday, November 10, 2014 at 11:39 AM
```

```
#Researcher Group's annotation obtained through Gene Expression Omnibus: GPL14112
################################################################################

#Read in data downloaded from Agilent Website
agilent <- read.delim("/Users/russell/Desktop/USU␣Student␣Materials/Research␣
    Summer␣2014/Data/AllAnnotations\\019921_D_AA_20141001.txt"
                    , stringsAsFactors=F, fill=TRUE, sep=)
#Read in data from Gene Expression Omnibus
research <- Table(getGEO("GPL14112", GSEMatrix=TRUE))

#Some functions to be used later
quickView <- function (data) {
  #Function to Get General Info about Annotation from Sources
  #Function takes data frame as input
  #Make sure all missing data is "" not NAs
  data[is.na(data)] <- ""
  overview <- matrix(ncol=6)
  for (i in 1:length(colnames(data))) {
    colVec <- data[,i]
    rows <- length(colVec)
    annotated <- length(colVec[colVec !=""])
    duplicates <- sum(duplicated(colVec[colVec !=""]) | duplicated(colVec[colVec !
        =""], fromLast=TRUE))
    uniques <- annotated-duplicates
    p.annot <- round(annotated/rows,2)
    p.un.annot <- round(uniques/rows,2)
    p.annot.dup <- round((annotated-uniques)/annotated,2)
    overview <- rbind(overview, c(rows,annotated,p.annot, uniques, p.un.annot, p.
        annot.dup))
  }
  overview <- overview[-1,]
```

```
  row.names(overview) <- colnames(data)
  colnames(overview) <- c("rows","rows␣w\annot.","%␣rows␣w\annot.","rows␣w\nd.
      annot.","%␣rows␣w\nd.annot.","%␣annot.␣duplicated")
  overview
}
gncTranslate <- function (bimapLeft, bimapRight, gnc.translator, all.probe.ids) {
  #Function to merge two bimaps and give info about the merge
  #Make sure there are no missing entries ("" or NA) in either bimap
  #Make sure merger is a character column name common to both bimaps (gnc to use
      as translator)

  merged <- distinct(merge(bimapLeft, bimapRight, gnc.translator))
  colnames(merged) <- c('gnc','probe_id','go_id','go_domain')

  #Microarray chip coverage as intersection of unique probe_ids after merge over
      all unique probe_ids
  common <- intersect(unique(all.probe.ids), unique(merged$probe_id))
  chipC <- length(common)/length(unique(all.probe.ids))

  sets <- length(unique(merged$go_id))
  gncs <- length(unique(merged$gnc))
  probes <- length(unique(merged$probe_id))
  annotations <- distinct(select(merged, gnc, probe_id))
  annots <- dim(annotations)[1]

  myList <- list()
  myList[[1]] <- merged
  myList[[2]] <- annotations
  myList[[3]] <- data.frame("GOgncs"=gncs, "probeIDs"=probes, "Annotations"=annots
      ,
                            "GO␣IDs"=sets, "chipCoveragePotential"=chipC)
  names(myList) <- c("groupBuilder", "annotations", "performance")
```

```
  myList
}
groupBuilder <- function (groupBuilder) {
  x <- groupBuilder$go_id
  y <- groupBuilder[,c("gnc","probe_id")]
  uniques <- unique(x)
  groupList <- setNames(vector("list", length(unique(x))), uniques)
  finish <- length(uniques)
  for (i in 1:finish)
  {
    t <- grep(uniques[i], x)
    groupList[[i]] <- y[t,]
  }
  groupList
}
alignments <- function (gnc.seqs, probe.seqs, annotations, gnc, probe_id) {

  mergedSeqs <- merge(merge(annotations, gnc.seqs, by=gnc), probe.seqs, by=probe_
      id)

  typeChoice <- c("global", "local", "overlap", "global-local","local-global")
  submat <- nucleotideSubstitutionMatrix(match = 0, mismatch = -1, baseOnly =
      FALSE)
  mergedSeqs <- data.frame(mergedSeqs,"align_score"=NA)
  finish <- dim(mergedSeqs)[1]
  for (i in 1:finish) {
    mergedSeqs[i, 5] <- pairwiseAlignment(pattern = mergedSeqs$gnc.seq[i], subject
        = mergedSeqs$probe_id.seq[i],
                                          substitutionMatrix = submat, gapOpening = 0,
                                               gapExtension = 1,
                                          scoreOnly = TRUE, type=typeChoice[4])
  }
```

```
  mergedSeqs[,c('probe_id','gnc','align_score')]
}
annotator <- function(m) {
  data.frame(m, "annotations"=apply(m, 1, function(x) paste(x[1],x[2], sep="<MAP>"
      )))
}
smp <- function (groupList.annot, m.annot) {
  #Each element in groupsList.annot should have three columns "gnc","probe_id","
      annotations"
  #m.annot should have four columns named "gnc","probe_id","align_score","
      annotations"
  groups <- list()
  finish <- length(names(groupList.annot))
  for (k in 1:finish) {
    groups[[k]] <- data.frame(groupList.annot[[k]], "score"=NA)
    finish1 <- dim(groups[[k]])[1]
    for (i in 1:finish1) {
      t <- grep(groups[[k]]$annotations[i], m.annot$annotations, fixed=TRUE)
      groups[[k]]$score[i] <- max(m.annot$align_score[t])
    }
  }

  #order each gnc type by score
  gnc.ordered.groups <- list()
  probes.ordered.groups <- list()
  for (i in 1:finish) {
    gnc.ordered.groups[[i]] <- groups[[i]][order(groups[[i]]$gnc, groups[[i]]$
        score, decreasing=T),]
    probes.ordered.groups[[i]] <- groups[[i]][order(groups[[i]]$probe_id, groups[[
        i]]$score, decreasing=T),]
  }
  length(gnc.ordered.groups)
```

```
length(probes.ordered.groups)
gnc.ordered.groups[20]
probes.ordered.groups[20]

list.groups <- list()
for(i in 1:finish) {
  n.pro <- length(unique(gnc.ordered.groups[[i]]$probe_id))
  n.gnc <- length(unique(gnc.ordered.groups[[i]]$gnc))
  u.pro <- unique(gnc.ordered.groups[[i]]$probe_id)
  u.gnc <- unique(gnc.ordered.groups[[i]]$gnc)
  gnc.ordered <- gnc.ordered.groups[[i]]$gnc
  probes.ordered <- probes.ordered.groups[[i]]$probe_id

  if (n.pro > 1 & n.gnc > 1) {
    gnc.prefs <- vector(,n.pro)
    for (k in 1:n.gnc) {
      t <- grep(u.gnc[k], gnc.ordered)
      gnc.prefs <- cbind( gnc.prefs, c(t, rep(NA,n.pro-length(t))) )
    }
    gnc.prefs <- gnc.prefs[,-1]

    probe.prefs <- vector(,n.gnc)
    for (a in 1:n.pro) {
      t <- grep(u.pro[a], probes.ordered)
      probe.prefs <- cbind( probe.prefs, c(t, rep(NA,n.gnc-length(t))) )
    }
    probe.prefs <- probe.prefs[,-1]

    result <- daa(s.prefs=gnc.prefs, c.prefs=probe.prefs)$match.mat
    colnames(result) <- u.pro
    list.groups[[i]] <- rownames(which(t(result)==TRUE, arr.ind = TRUE))
  } else {
```

```
      list.groups[[i]] <- as.vector(gnc.ordered.groups[[i]]$probe_id)
    }
  }
  list.groups
}

#####################################################################
#####SELECT GENE NAMING CONVENTION (GNC) TO USE AS TRANSLATOR
#Get overview info about annotation to find and select 'sensible' gnc-translator
    candidates.
#####################################################################
quickView(agilent)
quickView(research)

#########################################################
#GenbankAccession as gnc-translator (source:agilent)
#########################################################
#Select version of biomart (see biomart website for list of available archives)
#host="may20015 portion corresponds to ensembl version 80 (current version)
#Select organism with dataset="" option. To see possible choices use:
#listDatasets(ensembl_80)
ensembl_80_oaries <- useMart(biomart="ENSEMBL_MART_ENSEMBL", host="may2015.archive
    .ensembl.org",
                          path="/biomart/martservice", dataset="oaries_gene_
                              ensembl")

#Query biomart to get gnc-go bimap
#dim(listAttributes(ensembl_80_oaries))
#listAttributes(ensembl_80_oaries)[1:100,]
#listFilters(ensembl_80_oaries)
embl.go <- getBM(attributes=c('embl', 'go_id', 'namespace_1003'),
                filters="with_go_id", values=TRUE, mart = ensembl_80_oaries)
```

```
embl.go <- filter(embl.go, embl != "", go_id != "", namespace_1003 != "",
    namespace_1003=="biological_process")

#Get from source probeID-gnc bimap
#Make sure that gnc translator has common character name in both bimaps
probe.embl <- filter(select(agilent, ProbeID, GenbankAccession), ProbeID != "",
    GenbankAccession != "")
probe.embl <- select(probe.embl, ProbeID, embl=GenbankAccession)

#Use gncTranslate function to estimate gnc-translator quality
#Larger numbers in GO.IDs and chipCoveragePotential is better.
agilent.gnc.GB_ACC <- gncTranslate(probe.embl,embl.go,"embl",agilent$ProbeID)
#names(agilent.gnc.GB_ACC)
#head(agilent.gnc.GB_ACC$groupBuilder)
#head(agilent.gnc.GB_ACC$annotations)
agilent.gnc.GB_ACC$performance

###############################################
#Unigene as gnc-translator (source:agilent)
###############################################
#Query biomart to get gnc-go bimap
#dim(listAttributes(ensembl_80_oaries))
#listAttributes(ensembl_80_oaries)[1:100,]
#listFilters(ensembl_80_oaries)
unigene.go <- getBM(attributes=c('unigene', 'go_id', 'namespace_1003'),
                 filters="with_unigene", values=TRUE, mart = ensembl_80_oaries)
#Include distinct rows of row combined queries then filter rows to specified
    domain and get rid of missings
unigene.go <- filter(unigene.go, unigene != "", go_id != "", namespace_1003 != "",
     namespace_1003=="biological_process")

#Get from source probeID-gnc bimap
```

```
#Make sure that gnc translator has common character name in both bimaps
probe.unigene <- filter(select(agilent, ProbeID, UniGeneID), ProbeID != "",
    UniGeneID != "")
probe.unigene <- select(probe.unigene, ProbeID, unigene=UniGeneID)

#Use gncTranslate function to estimate gnc-translator quality
#Larger numbers in GO.IDs and chipCoveragePotential is better.
agilent.gnc.unigene <- gncTranslate(probe.unigene,unigene.go,"unigene",agilent$
    ProbeID)
#names(agilent.gnc.unigene)
#head(agilent.gnc.unigene$groupBuilder)
#head(agilent.gnc.unigene$annotations)
agilent.gnc.unigene$performance

##############################################
#Entrez as gnc-translator (source:agilent)
##############################################
#Query biomart to get gnc-go bimap
#dim(listAttributes(ensembl_80_oaries))
#listAttributes(ensembl_80_oaries)[1:100,]
#listFilters(ensembl_80_oaries)
entrez.go <- getBM(attributes=c('entrezgene', 'go_id', 'namespace_1003'),
                filters="with_entrezgene", values=TRUE, mart = ensembl_80_oaries)
#Include distinct rows of row combined queries then filter rows to specified
    domain and get rid of missings
entrez.go <- filter(entrez.go, entrezgene != "", go_id != "", namespace_1003 != ""
    , namespace_1003=="biological_process")

#Get from source probeID-gnc bimap
#Make sure that gnc translator has common character name in both bimaps
agilent$EntrezGeneID[is.na(agilent$EntrezGeneID)] <- "" #Careful with entrez in
    agilent. Missing data are NAs not "".
```

```
probe.entrez <- filter(select(agilent, ProbeID, EntrezGeneID), ProbeID != "",
    EntrezGeneID != "")
#Make sure that gnc translator has common character name in both bimaps
probe.entrez <- select(probe.entrez, ProbeID, entrezgene=EntrezGeneID)

#Use gncTranslate function to estimate gnc-translator quality
#Larger numbers in GO.IDs and chipCoveragePotential is better.
agilent.gnc.entrez <- gncTranslate(probe.entrez,entrez.go,"entrezgene", agilent$
    ProbeID)
#names(agilent.gnc.entrez)
#head(agilent.gnc.entrez$groupBuilder)
#head(agilent.gnc.entrez$annotations)
agilent.gnc.entrez$performance

######################################################################
#RefSeq as gnc-translator (source:agilent)
#This is an example of multiple types of gncs being classified as one by a
    manufactuerer.
#Agilent has refseq as one gnc but biomart has three for refseq.
#To handle this issue, multiple queries to biomart can be made to increase
    annotation
#coverage.
######################################################################
#Query biomart to get gnc-go bimap
#dim(listAttributes(ensembl_80_oaries))
#listAttributes(ensembl_80_oaries)[1:100,]
#listFilters(ensembl_80_oaries)
refseq_mrna.go <- getBM(attributes=c('refseq_mrna', 'go_id', 'namespace_1003'),
                values=TRUE, mart = ensembl_80_oaries)
refseq_mrna_predicted.go <- getBM(attributes=c('refseq_mrna_predicted', 'go_id', '
    namespace_1003'),
                    values=TRUE, mart = ensembl_80_oaries)
```

```
refseq_ncrna_predicted.go <- getBM(attributes=c('refseq_ncrna_predicted', 'go_id',
    'namespace_1003'),
                                values=TRUE, mart = ensembl_80_oaries)
#Make the multiple biomart queries one
colnames(refseq_mrna.go) <- c("refseq",'go_id','namespace_1003')
colnames(refseq_mrna_predicted.go) <- c("refseq",'go_id','namespace_1003')
colnames(refseq_ncrna_predicted.go) <- c("refseq",'go_id','namespace_1003')
#Include distinct rows of row combined queries then filter rows to specified
    domain and get rid of missing
refseq.go <- distinct(rbind(refseq_mrna.go,refseq_mrna_predicted.go,refseq_ncrna_
    predicted.go))
refseq.go <- filter(refseq.go, refseq != "", go_id != "", namespace_1003 != "",
    namespace_1003=="biological_process")

#Get from source probeID-gnc bimap
#Make sure that gnc translator has common character name in both bimaps
probe.refseq <- filter(select(agilent, ProbeID, RefSeqAccession), ProbeID != "",
    RefSeqAccession != "")
probe.refseq <- select(probe.refseq, ProbeID, refseq=RefSeqAccession)

#Use gncTranslate function to estimate gnc-translator quality
#Larger numbers in GO.IDs and chipCoveragePotential is better.
agilent.gnc.refseq <- gncTranslate(probe.refseq,refseq.go,"refseq", agilent$
    ProbeID)
#names(agilent.gnc.refseq)
#head(agilent.gnc.refseq$groupBuilder)
#head(agilent.gnc.refseq$annotations)
agilent.gnc.refseq$performance

################################################################################
#The research file has a gnc titled 'Primary Accession. This gnc actually is
#a composite of other gncs in the file, many of which are GB_ACCs.
```

```
#GB_ACC as gnc-translator (source:research)
################################################################################
#Query biomart to get gnc-go bimap has alread been done for emble (genbank) ID

#Get from source probeID-gnc bimap
#Make sure that gnc translator has common character name in both bimaps
probe.embl <- filter(select(research, ID, GB_ACC), ID != "", GB_ACC != "")
probe.embl <- select(probe.embl, ID, embl=GB_ACC)

#Use gncTranslate function to estimate gnc-translator quality
#Larger numbers in GO.IDs and chipCoveragePotential is better.
research.gnc.embl <- gncTranslate(probe.embl,embl.go,"embl", research$ID)
#names(research.gnc.embl)
#head(research.gnc.embl$groupBuilder)
#head(research.gnc.embl$annotations)
research.gnc.embl$performance

###########################################
#ORF as gnc-translator (source:research)
#ORF is not a gnc, but ORF contains majority
#HGNC symbols gnc and this was used.
###########################################
#Query biomart to get gnc-go bimap
#dim(listAttributes(ensembl_80_oaries))
#listAttributes(ensembl_80_oaries)[1:100,]
#listFilters(ensembl_80_oaries)
hgnc_symbol.go <- getBM(attributes=c('hgnc_symbol', 'go_id', 'namespace_1003'),
                    values=TRUE, mart = ensembl_80_oaries)
#Include distinct rows of row combined queries then filter rows to specified
    domain and get rid of missings
hgnc_symbol.go <- filter(hgnc_symbol.go, hgnc_symbol != "", go_id != "", namespace
    _1003 != "", namespace_1003=="biological_process")
```

```
#Get from source probeID-gnc bimap
#Make sure that gnc translator has common character name in both bimaps
probe.hgnc_symbol <- filter(select(research, ID, ORF), ID != "", ORF != "")
probe.hgnc_symbol <- select(probe.hgnc_symbol, ID, hgnc_symbol=ORF)

#Use gncTranslate function to estimate gnc-translator quality
#Larger numbers in GO.IDs and chipCoveragePotential is better.
agilent.hgnc_symbol <- gncTranslate(probe.hgnc_symbol,hgnc_symbol.go,"hgnc_symbol"
    ,research$ID)
#names(agilent.hgnc_symbol)
#head(agilent.hgnc_symbol$groupBuilder)
#head(agilent.hgnc_symbol$annotations)
agilent.hgnc_symbol$performance

##########################################################################
#Take best gnc-translator use SMP procedure to build GO Gene Set List
##########################################################################

#Local-Global Alignments to rank annotations within each unique go term
#First guild gene group sets
#Takes about 3-4 min.
groupList <- groupBuilder(agilent.hgnc_symbol$groupBuilder)

#Get Sequences for unique gnc and unique prob with biomart query
#Takes about 2-3 min.
gnc.seqs <- getBM(attributes=c('cdna','hgnc_symbol'),
                  filters="hgnc_symbol", values=unique(agilent.hgnc_symbol$
                      annotations$gnc),
                  mart = ensembl_80_oaries)
probe.seqs <- read.delim("/Users/russell/Desktop/USU␣Student␣Materials/Research␣
    Summer␣2014/Data/SequenceList\\019921_D_SequenceList_20141001.txt"
```

```
                    , stringsAsFactors=F, fill=TRUE, sep=)

#Change column names appropriately (probe identifier and gnc columns should have
    same names across bimaps)
colnames(gnc.seqs) <- c("gnc.seq","gnc")
colnames(probe.seqs) <- c("probe_id","probe_id.seq")
colnames(agilent.hgnc_symbol$annotations) <- c("gnc","probe_id")

#Sequence alignments for each unique combination of probe and gnc seqs.
#Some annotations will have different combinations of probe and gnc seqs, but
#they are few and this will be handled later in the smp function by picking the
#max alignment score for duplicate annotations.
#Sequences are used to get align_scores, but are not returned.
#Takes about 3-4 min.
agilent.hgnc_symbol.alignments <- alignments(gnc.seqs, probe.seqs, agilent.hgnc_
    symbol$annotations,
          colnames(agilent.hgnc_symbol$annotations)[1], colnames(agilent.hgnc_
              symbol$annotations)[2])
#dim(agilent.hgnc_symbol.alignments)

#Matrix of containing bimap where rows are annotations with align_scores for each
    row
#Be sure to put columns in appropriate order to match annotations in groupList
#(returned by groupBuilder.)
m <- agilent.hgnc_symbol.alignments[,c("gnc", "probe_id", "align_score")]

#The annotator function provides one annotation for each row (through paste
    function) and
#will help with computation time when matching later.
m.annot <- annotator(m)
groupList.annot <- lapply(groupList, annotator)
```

```
#Function to grab preferences, rank them for each element and use them in
#Gale-Shapley's Deferred Acceptance Algorithm (daa) also known as the stable
#marriage algorithm for the stable marriage problem (smp).
#Takes about #4-5 min.
list.groups <- smp(groupList.annot, m.annot)
#list.groups[1:10]

list.groups.oaries <- list.groups
#Put GO Terms back as names of list
names(list.groups.oaries) <- names(groupList)

#The hierarchical structure of GO necessitates getting the offspring GO terms and
    including them in the list
offspring <- get("as.list", pos = "package:AnnotationDbi")(get(paste("GO", "BP", "
    OFFSPRING", sep = "")))
full.list.groups.oaries <- sapply(1:length(offspring), function(x) fillInList(list
    .groups.oaries[[names(offspring[x])]],
                                                                    names(offspring)
                                                                        [x],
                                                                        offspring,
                                                                        list.groups.
                                                                        oaries))
names(full.list.groups.oaries) <- names(offspring)

#Only include gene sets with at least one gene
final.list.groups.oaries <- full.list.groups.oaries[sapply(full.list.groups.oaries
    , length)!=0]

#Check list for any duplicated probeIDs within each GO gene set
sum(unlist(lapply(final.list.groups.oaries, function (x) sum(duplicated(x)))))

#Save with meaningful name.
```

```
#Total procedure takes less than 15 min.
save(final.list.groups.oaries, file="/Users/russell/Desktop/USU␣Student␣Materials/
    Research␣Summer␣2014/Thesis/R␣Code/final.list.groups.oaries")
load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/final.list.groups.oaries")

####################
#mvGST Analysis
####################
#Get the expression data#
EsetGSE52888 <- getGEO("GSE52888",GSEMatrix=TRUE)
#It reads in as a list of length one. Point to the ExpressionSet and save as an
    object
#so that it can be used in the global environment.
EsetGSE52888 <- EsetGSE52888$GSE52888_series_matrix.txt.gz

#Get probeIDs associated with GSE52888
probe.ids <- getGEO("GPL10427",GSEMatrix=FALSE)
probe.ids <- Table(probe.ids)[,8]
probe.ids <- as.character(probe.ids)

#Some functions used to explore an expression set object
#class(EsetGSE52888)
# featureNames(EsetGSE52888)[1:5]
# fvarLabels(EsetGSE52888)
#sampleNames(EsetGSE52888)
#varLabels(EsetGSE52888)

#####################################################################
#Limma\eBayes to test DE among genes across contrasts of interest
#Results in a matrix of p-values
#####################################################################
```

```
#Extract matrix of expression values
mat <- exprs(EsetGSE52888)
#Name columns appropriately where each column identifys a particular sample
#C:Control
#BO:Brachiocephalic Occlusion
#EBO:Estradiol Plus Brachiocephalic Occlusion
#E:Estradiol
colnames(mat) <- c("C","C","C","C",
                   "BO","BO","BO","BO",
                   "EBO","EBO","EBO","EBO",
                   "E","E","E","E")
#Define treatments, fit linear model
Treatment <- as.factor(c("C","C","C","C","BO","BO","BO","BO",
                         "EBO","EBO","EBO","EBO","E","E","E","E"))
#Define design matrix
design <- model.matrix(~0+Treatment)
colnames(design) <- c('C','BO','EBO','E')
#Fit linear model
fit <- lmFit(mat, design)

#Test 1st contrast
contrast1 <- makeContrasts(BO-C, levels=design)
fit1 <- contrasts.fit(fit, contrast1)
final.fit1 <- eBayes(fit1)
top1 <- topTableF(final.fit1, adjust.method="BH", n=nrow(mat), sort.by="none")

#Test 2nd contrast
contrast2 <- makeContrasts(EBO-C, levels=design)
fit2 <- contrasts.fit(fit, contrast2)
final.fit2 <- eBayes(fit2)
top2 <- topTableF(final.fit2, adjust.method="BH", n=nrow(mat),sort.by="none")
```

```
#Test 3rd contrast
contrast3 <- makeContrasts(E-C, levels=design)
fit3 <- contrasts.fit(fit, contrast3)
final.fit3 <- eBayes(fit3)
top3 <- topTableF(final.fit3, adjust.method="BH", n=nrow(mat),sort.by="none")

#Prepare matrix of p-values
pvals.study2 <- cbind(top1$P.Value, top2$P.Value, top3$P.Value)
colnames(pvals.study2) <- c("BO-C","EBO-C","E-C")
rownames(pvals.study2)<- probe.ids

head(pvals.study2)

save(pvals.study2, file="/Users/russell/Desktop/USU␣Student␣Materials/Research␣
    Summer␣2014/Thesis/R␣Code/pvals.study2")

######################
#Study 1 Comparisons
######################
library(graphics)
library(Hmisc)
library(mvGST)
library(dplyr)

my.colors <- c("#66c2a5", "#fc8d62", "#8da0cb")

load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/profileTable.btaurus")
load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/biomart.btaurus")
```

```
load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/affy.btaurus")
load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/pvals.study1")

annot <- read.csv(file="/Users/russell/Desktop/USU␣Student␣Materials/Research␣
    Summer␣2014/Data/Bovine/Bovine.na35.annot.csv",head=TRUE,sep=",", skip=19)
total.probes <- length(unique(annot$Probe.Set.ID))

#Unique GO identifiers in each list
y <- list()
y[[1]] <- unique(names(biomart.btaurus))
y[[2]] <- unique(names(affy.btaurus))
y[[3]] <- unique(names(profileTable.btaurus))

#Unique genes in each list
x <- list()
x[[1]] <- unique(unlist(biomart.btaurus))
x[[2]] <- unique(unlist(affy.btaurus))
x[[3]] <- unique(unlist(profileTable.btaurus))

#Unique GO gene sets
w <- list()
w[[1]] <- unique(biomart.btaurus)
w[[2]] <- unique(affy.btaurus)
w[[3]] <- unique(profileTable.btaurus)

############################
#Compare list performace
############################
pdf("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/
    figures/listPerformancePlot.pdf")
```

```
plot(y=sapply(y, length), x=sapply(x, length)/total.probes, pch=c(21,22,23),
    bg=my.colors, xlab="CHIP␣COVERAGE", ylab="UNIQUE␣GO␣IDS", cex=2)
legend(0.42, 9400, c("Biomart", "Affymetrix", "mvGST"), pt.bg=my.colors,
      pch=c(21,22,23), bty="n", cex=1.2, pt.cex=2)
dev.off()

##############################
#Compare lists to each other
##############################

#GO Term Identifiers
ycomp <- vector()
ycomp[1] <- length(intersect(y[[1]], y[[2]]))
ycomp[2] <- length(unique(setdiff(union(y[[1]], y[[2]]), intersect(y[[1]], y[[2]])
    )))
ycomp[3] <- length(intersect(y[[1]], y[[3]]))
ycomp[4] <- length(unique(setdiff(union(y[[1]], y[[3]]), intersect(y[[1]], y[[3]])
    )))
ycomp[5] <- length(intersect(y[[2]], y[[3]]))
ycomp[6] <- length(unique(setdiff(union(y[[2]], y[[3]]), intersect(y[[2]], y[[3]])
    )))
ycomp[7] <- length(intersect(intersect(y[[1]], y[[2]]), y[[3]]))
ycomp[8] <- length(unique(setdiff(union(y[[1]], y[[2]]), y[[3]]),
                     intersect(intersect(y[[1]], y[[2]]), y[[3]])))

y.inters <- c(ycomp[1],ycomp[3],ycomp[5],ycomp[7])
y.not.inters <- c(ycomp[2],ycomp[4],ycomp[6],ycomp[8])
y.percent.shared <- y.inters/(y.inters+y.not.inters)

#Unique Gene Identifiers
xcomp <- vector()
xcomp[1] <- length(intersect(x[[1]], x[[2]]))
```

```
xcomp[2] <- length(unique(setdiff(union(x[[1]], x[[2]]), intersect(x[[1]], x[[2]])
    )))
xcomp[3] <- length(intersect(x[[1]], x[[3]]))
xcomp[4] <- length(unique(setdiff(union(x[[1]], x[[3]]), intersect(x[[1]], x[[3]])
    )))
xcomp[5] <- length(intersect(x[[2]], x[[3]]))
xcomp[6] <- length(unique(setdiff(union(x[[2]], x[[3]]), intersect(x[[2]], x[[3]])
    )))
xcomp[7] <- length(intersect(intersect(x[[1]], x[[2]]), x[[3]]))
xcomp[8] <- length(unique(setdiff(union(x[[1]], x[[2]]), x[[3]]),
                      intersect(intersect(x[[1]], x[[2]]), x[[3]])))

x.inters <- c(xcomp[1],xcomp[3],xcomp[5],xcomp[7])
x.not.inters <- c(xcomp[2],xcomp[4],xcomp[6],xcomp[8])
x.percent.shared <- x.inters/(x.inters+x.not.inters)

#GO Gene Sets
#Takes approx. to run, best to save results
wcomp <- vector()
wcomp[1] <- length(intersect(w[[1]], w[[2]]))
wcomp[2] <- length(unique(setdiff(union(w[[1]], w[[2]]), intersect(w[[1]], w[[2]])
    )))
wcomp[3] <- length(intersect(w[[1]], w[[3]]))
wcomp[4] <- length(unique(setdiff(union(w[[1]], w[[3]]), intersect(w[[1]], w[[3]])
    )))
wcomp[5] <- length(intersect(w[[2]], w[[3]]))
wcomp[6] <- length(unique(setdiff(union(w[[2]], w[[3]]), intersect(w[[2]], w[[3]])
    )))
wcomp[7] <- length(intersect(intersect(w[[1]], w[[2]]), w[[3]]))
wcomp[8] <- length(unique(setdiff(union(w[[1]], w[[2]]), w[[3]]),
                      intersect(intersect(w[[1]], w[[2]]), w[[3]])))
```

```
save(wcomp, file="/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣
    2014/Thesis/R␣Code/wcomp")
load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/wcomp")

w.inters <- c(wcomp[1],wcomp[3],wcomp[5],wcomp[7])
w.not.inters <- c(wcomp[2],wcomp[4],wcomp[6],wcomp[8])
w.percent.shared <- w.inters/(w.inters+w.not.inters)

concordance <- cbind(y.percent.shared,x.percent.shared, w.percent.shared)
colnames(concordance) <- c("GOID","GeneID","GeneSets")
concordance <- arrange(as.data.frame(concordance), desc(GOID))

pdf("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/
    figures/listConcord.pdf")
dotchart2(concordance[,"GOID"], labels= c("Biomart␣␣␣\nAffy␣␣␣␣",
                                    "Affy␣␣␣␣␣␣\nmvGST␣",
                                    "Biomart␣␣␣\nmvGST␣",
                                    "Biomart␣␣␣\nAffy␣␣␣␣␣␣\nmvGST␣"),
         pch=21, xlim=c(0,1), dotsize=1.2,
         bg=rep(my.colors[1],4), xaxis=FALSE, cex.lab=1.5)
axis(side = 1, at = c(0,.25,.50,.75,1), labels=c("0",".25",".50",".75","1"),
     line=0.5)
title(xlab="Proportion␣Concordant", line=3, cex=1)
dotchart2(concordance[,"GeneID"], pch=22, dotsize=1.2, add=TRUE,
         bg=rep(my.colors[2],4))
dotchart2(concordance[,"GeneSets"], pch=23, dotsize=1.2, add=TRUE,
         bg=rep(my.colors[3],4))
legend(.08, 1.5, legend=c("GO␣IDs", "Gene␣IDs", "Gene␣Sets"),
       pt.bg=my.colors, pch=c(21,22,23), bty="n", cex=0.8, pt.cex=1)
dev.off()
```

```
########################
#Compare mvGST Results
########################
results <- list()
results[[1]] <- profileTable(pvals=pvals.study1, list.groups=profileTable.btaurus)
results[[2]] <- profileTable(pvals=pvals.study1, list.groups=affy.btaurus)
results[[3]] <- profileTable(pvals=pvals.study1, list.groups=biomart.btaurus)

#Day 12
p.n.12 <- cbind(results[[1]]$results.table[,"P-Day12.NP"],
               results[[2]]$results.table[,"P-Day12.NP"],
               results[[3]]$results.table[,"P-Day12.NP"])
percent.pn12 <- cbind(p.n.12[,1]/sum(p.n.12[,1]),
                   p.n.12[,2]/sum(p.n.12[,2]),
                   p.n.12[,3]/sum(p.n.12[,3]))
percent.pn12
pdf("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/
    figures/resultsDay12.pdf")
dotchart2(percent.pn12[1,], labels= c("mvGST",
                           "Affy",
                           "Biomart"),
         pch=21, xlim=c(0,1), dotsize=1.2,
         bg=rep(my.colors[1],4), xaxis=FALSE)
axis(side = 1, at = c(0,.25,.5,.75,1),
     labels=c("0",".25",".5",".75","1"),
     line=0.5)
title(xlab="Proportion␣of␣GO␣Gene␣Sets", line=3, cex=1)
dotchart2(percent.pn12[2,], pch=22, dotsize=1.2, add=TRUE,
         bg=rep(my.colors[2],4))
dotchart2(percent.pn12[3,], pch=23, dotsize=1.2, add=TRUE,
         bg=rep(my.colors[3],4))
legend(.1, 1.4, legend=c("Not␣Significant", "More␣Active", "Less␣Active"),
```

```
      pt.bg=my.colors, pch=c(21,22,23), bty="n", cex=0.8, pt.cex=1)
dev.off()

#Day 14
p.n.14 <- cbind(results[[1]]$results.table[,"P-Day14.NP"],
              results[[2]]$results.table[,"P-Day14.NP"],
              results[[3]]$results.table[,"P-Day14.NP"])
percent.pn14 <- cbind(p.n.14[,1]/sum(p.n.14[,1]),
                    p.n.14[,2]/sum(p.n.14[,2]),
                    p.n.14[,3]/sum(p.n.14[,3]))
percent.pn14
pdf("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/
    figures/resultsDay14.pdf")
dotchart2(percent.pn14[1,], labels= c("mvGST",
                              "Affy",
                              "Biomart"),
         pch=21, xlim=c(0,1), dotsize=1.2,
         bg=rep(my.colors[1],4), xaxis=FALSE)
axis(side = 1, at = c(0,.25,.50, .75, 1),
     labels=c("0",".25",".50",".75","1"),
     line=0.5)
title(xlab="Proportion␣of␣GO␣Gene␣Sets", line=3, cex=1)
dotchart2(percent.pn14[2,], pch=22, dotsize=1.2, add=TRUE,
         bg=rep(my.colors[2],4))
dotchart2(percent.pn14[3,], pch=23, dotsize=1.2, add=TRUE,
         bg=rep(my.colors[3],4))
legend(.1, 1.4, legend=c("Not␣Significant", "More␣Active", "Less␣Active"),
      pt.bg=my.colors, pch=c(21,22,23), bty="n", cex=0.8, pt.cex=1)
dev.off()

##################
#Study 2 Results
```

```
##################
agilent <- read.delim("/Users/russell/Desktop/USU␣Student␣Materials/Research␣
    Summer␣2014/Data/AllAnnotations\\019921_D_AA_20141001.txt"
                    , stringsAsFactors=F, fill=TRUE, sep=)
load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/pvals.study2")
load("/Users/russell/Desktop/USU␣Student␣Materials/Research␣Summer␣2014/Thesis/R␣
    Code/final.list.groups.oaries")


profileTable(pvals.study2, list.groups=final.list.groups.oaries)
```