# MICROARRAY TOOLS AND ANALYSIS METHODS TO BETTER CHARACTERIZE BIOLOGICAL NETWORKS

by

Christopher C. Overall

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Bioinformatics and Computational Biology

Charlotte

2012

Approved by:

_____
Dr. Jennifer Weller

_____
Dr. Xiuxia Du

_____
Dr. Anthony Fodor

_____
Dr. ZhengChang Su

_____
Dr. Jing Xiao

# ABSTRACT

CHRISTOPHER C. OVERALL. Microarray tools and analysis methods to better characterize biological networks.
(Under the direction of DR. JENNIFER WELLER)

To accurately model a biological system (e.g. cell), we first need to characterize each of its distinct networks. While omics data has given us unprecedented insight into the structure and dynamics of these networks, the associated analysis routines are more involved and the accuracy and precision of the experimental technologies not sufficiently examined. The main focus of our research has been to develop methods and tools to better manage and interpret microarray data. How can we improve methods to store and retrieve microarray data from a relational database? What experimental and biological factors most influence our interpretation of a microarray's measurements? By accounting for these factors, can we improve the accuracy and precision of microarray measurements? It's essential to address these last two questions before using 'omics data for downstream analyses, such as inferring transciption regulatory networks from microarray data. While answers to such questions are vital to microarray research in particular, they are equally relevant to systems biology in general.

We developed three studies to investigate aspects of these questions when using Affymetrix expression arrays. In the first study, we develop the Data-FATE framework to improve the handling of large scientific data sets. In the next two studies, we developed methods and tools that allow us to examine the impact of physical and technical factors known or suspected to dramatically alter the interpretation of a microarray experiment. In the second study, we develop ArrayInitiative – a tool that simplifies the process of creating custom CDFs – so that we can easily re-design the array specifications for Affymetrix 3' IVT expression arrays. This tool is essential

for testing the impact of the various factors, and for making the framework easy to communicate and re-use. We then use ArrayInitiative in a case study to illustrate the impact of several factors known to distort microarray signals. In the third study, we systematically and exhaustively examine the effect of physical and technical factors – both generally accepted and novel – on our interpretation of dozens of experiments using hundreds of E. coli Affymetrix microarrays.

# DEDICATION

*For my parents*

# ACKNOWLEDGMENTS

First, I'd like to thank my advisor Jennifer Weller for her guidance, support, encouragement and patience over the years. She is both a fierce critic and tireless ally, and this dissertation would not have been possible without her. I would also like to thank Andrew Carr, who has been a great collaborator and friend, and who, along with my advisor, has inspired me to think big during many lively discussions over food and spirits. I'd like to thank my committee members – Xiuxia Du, Anthony Fodor, ZhengChang Su and Jing Xiao – for their advice and comments. And of course, I need to acknowledge the financial support from the Department of Bioinformatics and Genomics and my GAANN fellowship.

Next, I'd like to thank my labmates and intellectual siblings - Cristina Baciu, Kevin Thompson, and most especially, Saeed Khoshnevis. Saeed has been a labmate, roommate and great friend since I've been at UNCC. While I'm excited to move on, I'll miss him dearly.

I have had an amazing group of friends, both outside and inside the graduate school, while attending GMU and UNCC. I won't list all of them, for fear of forgetting anyone, but they know who they are. All of them have been a source of fun, conversation, comfort and inspiration, and they deserve my deepest thanks. I especially want to thank my friend Dave, who's been a friend since we were children. Your friendship and continued excitement about my dissertation, even when I became weary, was instrumental to completing it.

Finally, I'd like to thank my parents, brother and sister for their love, support and encouragement - I love all of you. A special thanks to my parents, who have helped me in numerous, and often unacknowledged, ways.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1: INTRODUCTION

Advances in experimental and computational technology ushered in the current 'omics era, which is characterized by large-scale, high-throughput experiments, and with this wealth of data, it is now possible to start modeling and even building complete biological systems, known respectively as systems biology and synthetic biology. While giving us unprecedented insight into the dynamics of a cell, using the results from omics experiments poses numerous challenges for modern researchers, from storage and organization to scaling modeling environments so they can handle the inputs; the data sets are complex and massive, the associated analysis routines are more involved and the accuracy and precision of the experimental technologies are not sufficiently examined.

The main focus of our research has been to develop methods and tools to better manage and interpret microarray data. While past research focused most on correctly predicting strong clinical markers of a number of diseases, the ability to understand regulatory networks and underlying mechanisms of the diseases has been a longer range goal. Problems of importance to this research include: How can we improve methods to store and retrieve very large sets of microarray data from a relational database? What physical and technical factors most confound our interpretation of a microarray's measurements? Does deprecating confounded probes allow us to improve the accuracy and precision of microarray measurements? It is essential to address these last two questions before using 'omics data for downstream analyses, such as inferring transciption regulatory networks from microarray data. While microarray data first raised these questions, they are relevant to 'omics research in general.

## 1.1 Systems biology, synthetic biology and networks

Systems biology studies multi-component biological systems found in living cells, organisms and communities, seeking to understand how their complex interactions give rise to emergent functions and behaviors at multiple scales, the most fundamental questions and mysteries in biology [1, 2]. Pragmatically, we'd like to use this understanding to identify more robust biomarkers and better predict drug targets, using a systems-level understanding to promote efficacy and limit side effects. More idealistically, our goal is to understand mechanisms that underlie function, including redundancies and fault tolerances. Synthetic biology is the test bed for biological subsystems, using biotechnology to engineer biosimilar components, including systems not found in nature, creating novel functions and behaviors [3]. For example, the engineering of completely synthetic hydrohexitol nucleic acid polymers (HNAs) and highly modified polymerases carries the promise of creating non-toxic, non-degradable aptamers for binding proteins [4]. Synthetic biology offers the opportunity to build and test systems in isolation from the full complexity of their biological origin. Just as the findings in systems biology inform the design of synthetic biological systems, the failures and successes in synthetic biology inform our understanding of interactions in natural biological systems [3].

Complex systems can be represented as networks that are modeled as graphs [5]. The first step is to break down the system into autonomous, or nearly autonomous, subsystems, characterizing the participants, interactions, state parameters, and outside inputs. Systems biology uses this paradigm of interacting networks to represent many types of cellular systems, including the protein-protein interaction network [6], transcription regulatory network (TRN) [7, 8], signaling network [9, 10] and metabolic network [11, 12]. The network elements are not disconnected, but the specific focus helps identify points where networks modulate one another [13].

There are two major challenges to understanding and modeling a network: char-

acterizing its complete structure (static interactome) and modeling which parts are used under specific conditions (dynamic interactome)[14]. The static interactome includes the molecular components (nodes) and all of their possible interactions (edges) - basically, this is an hierarchical parts list [15]. The dynamic interactome describes how the network responds to internal and external signals (e.g. which subnetworks are active, how information propagates through the network, timing etc.). Generally, many data sets are integrated to infer the global structure of the network , while experiments based on time series or individual factors are used to understand the dynamics [14, 16]. It is also important to understand how the subnetworks interact with each other [17]. Since networks may interact at several points that shift depending on the state, massive amounts of experimental data must be generated across many environmental states and genetic conditions. This data must then be integrated appropriately in both spatial and temporal scales [18]. As a data-driven science, it is essential to capture the imperfections of measurement platforms so that integration is performed correctly. The preceding decade of microarray-intensive research has illustrated these problems well, from how sensor design influences outcomes to the integration challenges presented when read-out devices differ in sensitivity and specificity [19, 20, 21, 22]. The consequences of not removing suspect data are well-documented for clinical tests, but are less publicized with respect to networks [23, 24].

Most genomics experiments that aim to elucidate the dynamics of a system actually capture a series of static snapshots, with the aim of finding the common signature for a particular class of molecule over many cells, although there are also efforts to observe single cells using methods tuned to the time step of the molecular species [25]. As first expressed by the Functional Genomics Data (FGED) Society (formerly, the Microarray Gene Expression Databases [MGED] Society) in their MIAME standard [26], and later mimicked by many others [27, 28, 29] the data management system that can organize all of the information required to replicate the experiment is diffi-

cult to design, especially if a consistent model that yields analytical speed is the goal [30, 31]. A significant result of these early efforts was recognition that controlled, structured vocabularies, or ontologies, must be developed in parallel with the data management hardware and software before effective data integration or data mining could be achieved [32]. Not only are these resources needed for each type of network, but as we move towards integrating networks, additional systems must be developed [33, 34].

For the remainder of this chapter, we'll focus on transcription regulatory networks and the relevant experimental technologies used to characterize them.

## 1.2 E. coli and its TRN

Transcription regulatory networks are the foundation for all other biological networks, since proteins originate as transcripts and metabolites are created or moved around by proteins, while together proteins and metabolites modulate transcription; the modulation of transcript production is the initial regulatory network of cells. The most completely studied TRN is that of E. coli, and E. coli is the most completely understood biological system. This is true for many historical and practical reasons: studying prokaryotes is easier than studying eukaryotes since they lack alternative splicing, they have fewer genes and are amenable to genetic manipulation, and many are relatively easy to culture in the lab (especially E. coli). Since E. coli has been a model genetic and experimental organism for many years, and is used extensively in biotechnology, there is an enormous amount of 'omics data. In particular, many of the transcription factors and the units they regulate have been comprehensively characterized across many environmental states and genetic states. A large number of these datasets are in the public domain, and thus serve as a foundation for studies such as ours [35].

## 1.3 Characterizing TRNs with microarray data

To fully characterize the structure and dynamics of a TRN, you must monitor all of its nodes simultaneously, over many conditions and time points. The first 'omics platform to deliver this capability was the DNA microarray.

### DNA microarrays

The DNA microarray simultaneously assays all of the complementary nucleic acids in a target solution. Thus it is a popular experimental tool for carrying out many types of genomic and transcriptomic experiments, including gene expression profiling, detection of protein-DNA interactions, sequencing, comparative genomics, detection of single-nucleotide polymorphisms (SNPs) and copy number variation (CNV) and detection of alternative splicing. The arrays are constructed as two-dimensional grids of oligonucleotides ('probes'), each unique sequence is covalently affixed to a solid support at a specific location ('feature') via spotting or direct synthesis to reactive groups. Genomic features that we wish to assay may vary from a single nucleotide polymorphism (SNP) to units that are tens of thousands of nucleotides in length. Except for SNPs, for most features the probes are shorter than the corresponding targets. This presents an opportunity to sample the target multiple times, and some platforms are designed to accomplish that goal [36]. It is also possible to construct a sampling hierarchy based on co-location of elements on a molecule. As an example, in prokaryotes the genes on an operon may be co-transcribed. So there may be two probes that sample one gene, three probes that sample another, and the 5 probes together sample the operon. To define this sort of relationship, we use the term 'affiliated probes' in this dissertation, meaning any set of probes that measures the same contiguous nucleic acid target. An example of this is shown in Fig. 1.1.

Important distinctions between microarray platforms include the probe length and probe density on the array. Length allows greater sensitivity, but at the expense of specificity unless hybridization conditions can be tuned. The length of probes on

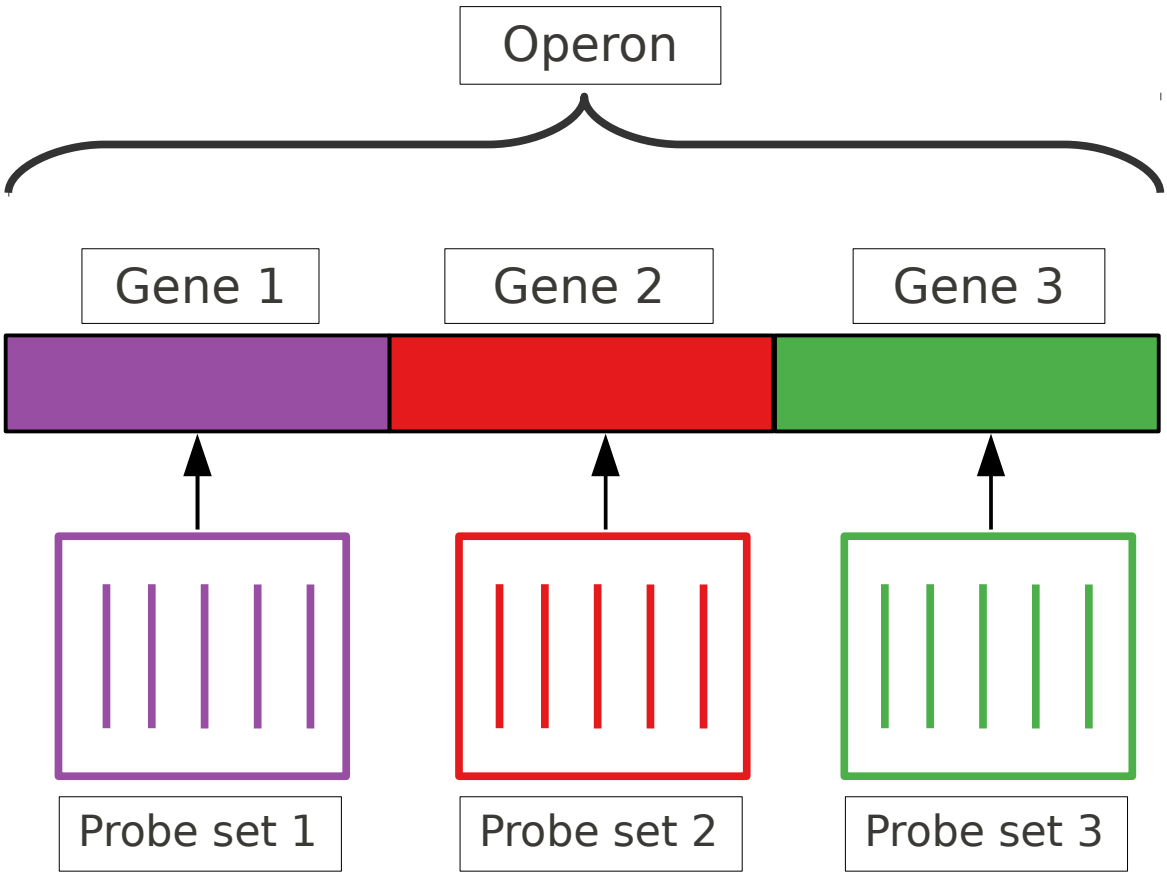


Figure 1.1: Example of affiliated probes.

the most widely used platform is 25nt (Affymetrix, [37]) and on currently marketed commerical platforms the maximum length is 70nt (Agilent, [38]). The 'probe density' can mean either the number of spots per array, which corresponds to the level of genome feature coverage, or to the concentration of the probe within a spot, which correlates to the percent of target that can be bound [39].

Although microarry platform designs differ, the basic procedure for running a microarray experiment is the same for all of them. When a solution of some labeled, purified cellular fraction is assayed against a microarray, a target will hybridize to any sufficiently complementary probes (intended or not); those not bound after the required reaction time are washed away. Target is labeled if this was not already done, and then the array is scanned, inducing the target-bound fluorophore to emit photons (signal), which are then captured and saved as an image file. For the computational scientist, the intensity of the pixels in this file is the starting point for estimating the concentration of each target molecule [39].

Affymetrix DNA microarrays

Affymetrix [37] produces the most widely used of the high-density DNA microarrays (called the GeneChip®platform). Their complexity results from the promiscuous placement of probes with respect to target elements and the multiplicity of probes per biological target. In this case, high probe density means both that there are hundreds of thousands to millions of simultaneous measurements to be considered and that the local concentration of probe is very high. The probes are synthesized in-situ, with relatively short lengths (25-33nt). Although the physical design of the arrays – a two-dimensional grid of probes – is identical to other types of DNA microarrays, the logical design differs significantly since multiple distinct probes (from 4 to 16 or more, depending on the specific array type) must be merged to report on any given feature. The advantage to such a design is the redundancy of the measurements: probe or target characteristics that fail to report accurately can be removed or reassigned while

leaving sufficient probes to faithfully report on the intended target.

These probes are grouped into sets, as shown in Fig. 1.2, which shows the logical design for 3' IVT expression arrays.

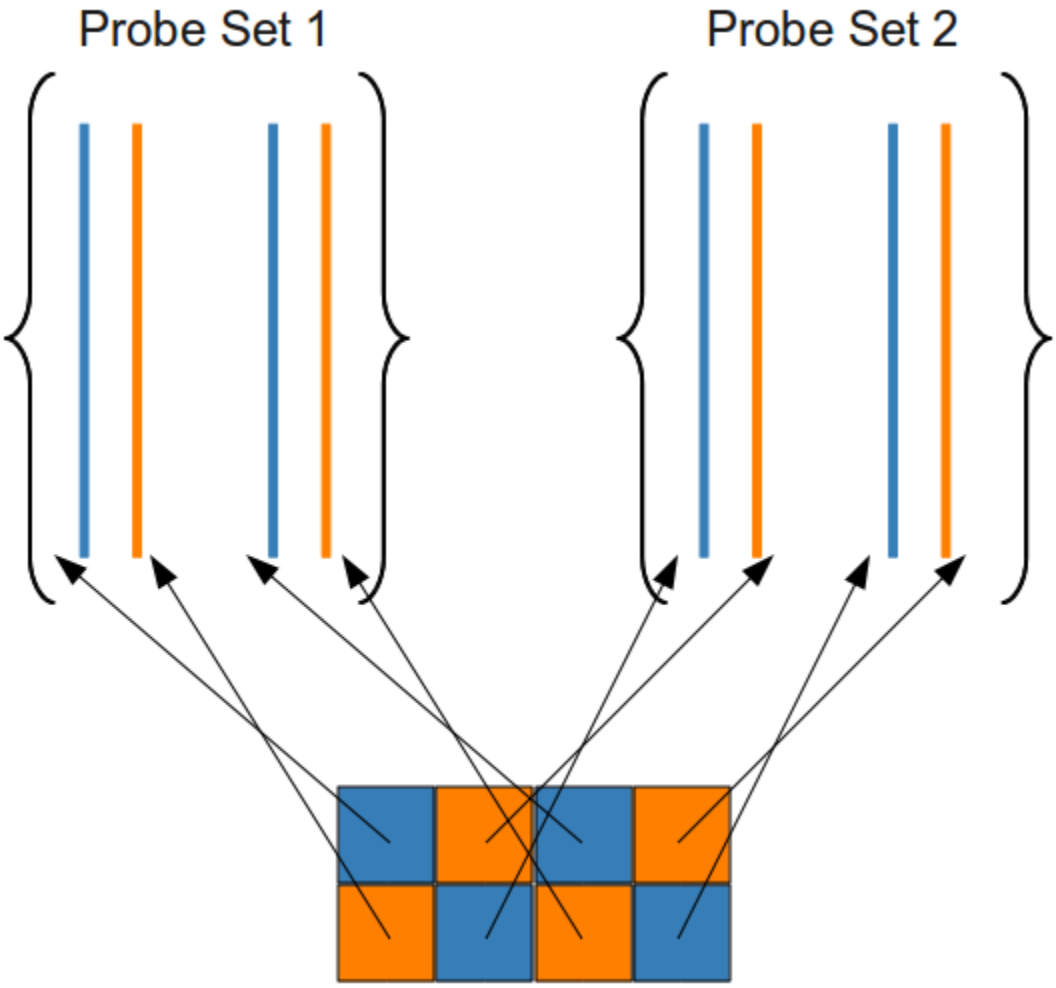


Figure 1.2: The logical design of an Affymetrix GeneChip®.

An Affymetrix-defined probe set is one level in the affiliated probe hierarchy which might discriminate variants of a transcript, exon, or SNP. The class of expression arrays is most relevant to our interest in studying TRNs. For several generations of the 3' IVT expression arrays, the basic measurement 'unit' was a probe pair, which included one perfect match (PM) probe and one mismatch (MM) probe, having a

single homomeric transversion at the central position relative to the PM [37]. For example, the E. coli antisense genome array, which will be the focus of Chapter 4 (ProbeSieve), consists of 141,629 probe pairs organized into 7,312 probe sets - 4,426 of them are designed for open reading frames (ORFs), while the remaining 2,886 target intergenic regions of the E. coli K12 genome. Most the of the probe sets have either 15 or 16 probe pairs.

Goals of a microarray experiment

Gene expression microarray experiments are performed to compare the transcriptomes produced by different biological conditions (e.g. healthy vs. diseased state) and to characterize the structure and dynamics of TRNs across time intervals, conditions and genetic backgrounds. A transcriptome is generally considered to be a description of the genes transcribed above a background level, but in the case of a prokaryote like E. coli, the transcriptome could be considered a description of the transcription units (TUs) produced above a background level, a distinction with a difference for multi-gene operons. Since regulation affects operons, an accurate inference of the TU level is our goal when describing TRNs.

To infer co-regulation of operons requires that the results of many experiments be compared, since two TUs may well both change without a common regulatory cause .Only if TUs consistently change levels, in the same direction and over many conditions, is it likely that co-regulation is occurring. Microarrays are valuable tools for characterizing the structure and dynamics of TRNs, as researchers can collect a large amount of experimental data across many environmental states and genetic conditions[40, 41].

## 1.4 Relevance of microarray data to E. coli TRN research

Although high-throughput sequencing (HTS) technologies [42] are starting to supplant microarrays as the predominant data-gathering platform for transcript levels, data from DNA microarrays will continue to be produced and used for the foresee-

able future [43]. In part this is because, as an established platform there are core labs producing high quality data and experimentalists well trained in producing the samples. With respect to analysis, the pipelines for data cleansing and data mining are far more mature for microarray data than for RNA-Seq data, and more experts have been trained to use them. Many E coli expression microarray data set are available in public repositories, which is not yet true for HTS data. In fact, NCBI, the manager of the Gene Expression Omnibus microarray data repository [44], has announced that it won't store HTS data because it's too resource-intensive. Because of its size and current lack of model standards, data sharing is more difficult with HTS data than for microarray data. Thus, expression microarray data remain a valuable resource for the discovery of TRNs.

## 1.5 Microarray data: storage, quality assessment, and integration

Despite 15 years of concentrated effort by a large group of experts, there remain many challenges associated with storing, managing and using large amounts of microarray data, particularly with respect to determining data quality. This is highlighted by a recent report in which data mining results based on microarray data were shown to be independent of the experimental variables [45].

In section 1.6, we discuss the essential components for storing, managing and using any type of experimental data, which includes building ontologies, developing databases and creating analytical workflows. This is the background for research described in Chapter 2 (Data-FATE). Once raw data has been properly organized it can be filtered to identify responses that arise due to the experimental variables and nothing else. There are well-known reproducibility problems associated with microarray data (section 1.7) and known contributing factors (section 1.8). Most analytical workflows require an array specification in order to process the filtered (cleansed) data. The review of current methods (sections 1.9 and 1.10) provides the context for Chapters 3 (ArrayInitiative) and 4 (ProbeSieve), which describe a tool for

re-defining probe sets and methods for determining groups of probes with correlated responses to be assigned to such groups.

## 1.6 Organizing 'omics data

The principle challenges when sharing any type of experimental data include finding mechanisms to communicate the meaning of the data and tools for disseminating the data so that it can be retrieved in useful parts as well as entire sets. Data models, database management systems, common vocabularies and shared workflows are all essential before a community can make use of a common resource. Although many innovative models and management software environments continue to be developed [46], the maturity of the relational data model and associated management software has led to their preeminence across the spectrum of biological databases [47]. Meaningful descriptions require a shared vocabulary. Logically structuring those vocabularies allows one to reason about relationships, as biologists demonstrated with the taxonomic tree of life. Thus ontologies have become an important organizing method for biological data. The development and testing of such ontologies has become a research focus in its own right, and includes the definition of entities, relationships, data models (e.g. database, exchange formats, etc) and data formats. Finally, to analyze your data consistently (read: arrive at reproducible results), you need to develop and use analytical workflows, with varying degrees of rigor.

With the dramatic increase in the volume and complexity of 'omics data (microarray, proteomics, next-gen sequencing), the scope of these challenges has increased abruptly. Here we'll discuss each of these challenges – ontologies, databases and analytical workflows – in general, and as they pertain to Affymetrix microarrays specifically. This section contains the relevant background for Chapter 2 - Data-FATE.

### 1.6.1 Ontologies

Humans constantly construct models of the universe around them, whether it be a process model or a data model. Process models describe a set of linked transforma-

tions that allow us to understand how something works and predict the outcome of new inputs or parameters e.g. creating a dynamical model of a transcription regulatory network [48, 49]. Data models (including ontologies) define the elements under scrutiny and all of the possible relationships that a process might use to link them. Not only is the data structured, but if the structure is a logical one then consistent representation of the data is assured. This is essential for the discovery of new relationships, which have often emerged when microarray data are used to characterize networks like TRNs [48]. Ontologies are also essential for data integration, which is one of the major challenges in Bioinformatics and systems biology [50, 18].

Definition of an ontology

An ontology – as defined in the information sciences – is a formal definition of concepts from a particular domain, and the relationship between those concepts. It is essentially a structure, or model, imposed on data in order to make sense of it, and to reason about it. Like the natural scientists' mathematical models, the ontology describes the participants in a physical process, and how those participants interact with each other.

More rigorously, a formal ontology consists of concepts, attributes and relations [51]. A concept is an object, such as a gene, that has at least one attribute (feature), such as genomic location. Concepts and their attributes are analagous to objects and their properties as defined in object-oriented programming. Concepts can range from general to specific and one concept can be a sub-concept of another (with the requirement that they share at least one attribute). Most biological ontologies have used the container ('is-a' and 'part-of') relationships only, although more active verbs are slowly being adopted by the Gene Ontology and the Systems Biology Ontologies [52]. For example, you could have a gene concept, with two subconcepts being prokyarotic gene and eukaryotic gene. Finally, an ontology defines relations between concepts; these are like the verbs and connectives in a sentence. For example, a gene

is 'expressed' as a gene product, either RNA or protein. In this case this relation is uni-directional, from gene to gene product, indicating a logical limit on the process that transforms one to the other. The easiest way to visualize an ontology is as a graph, where the nodes are concepts and the edges represents a relation between two concepts.

There are two types of ontologies: prescriptive and descriptive. While they both define concepts, attributes and relations, they differ in how they are constructed and used.

Prescriptive and descriptive ontologies

A *prescriptive ontology* (*shared ontology, top-down ontology, inductive ontology*) is one developed, and agreed upon, by the community of use (e.g. microarray researchers). This is a top-down approach where the concepts, attributes and relationships are prescribed by a group of experts - it's expected that all new data will conform to the structure of the ontology and that all researchers will use it without modification. While strict and rigid, they're essential for clear communication in the sciences. Having a specialized language for communicating in a particular domain ensures that domain experts are talking about the same things in the same way, with concepts and relationships that are precise and unambiguous . For example, most biologists will agree on the general definition of a gene, even if some attributes vary by sub-specialty. Placed within a formalized, logical structure, prescriptive ontologies also allow reasoning. If you're working within a well-established physical system or experimental procedure – where there's deep understanding about the processes and data – then a prescriptive ontology likely already exists or is straightforward to develop. Examples of prescriptive ontologies in Bioinformatics include the MGED ontology [26], Gene Ontology (GO) [53], the Systems Biology Ontology (SBO) [54], with its related Kinetic Simulation Algorithm Ontology (KiSAO), and Terminology for the Description of Dynamics (TEDDY)) [54].

Although prescriptive ontologies provide significant benefits, they also have significant overhead: since they are driven by community consensus, they are slow to update to new concepts and relationships, and as they become both large and complex it can be more difficult to test additions thoroughly. Especially in areas undergoing rapid expansion it is unlikely that everyone agrees on the meaning of the new data. As the ontology grows (or becomes bloated, depending on your perspective), researchers are often forced to use irrelevant terms, and overhead many refuse to accept. Some ontologies have been built with this in mind, with structure that allows pruning, so one retains only the most useful set of terms [55].

A *descriptive ontology* (*bottom-up, inductive*) is one built directly from what is known about the data: it describes the experimental factors that affect the data. If you're working in a frontier science, where the understanding of system components and relationships is dynamic, a prescriptive ontology is at best going to be under development. Descriptive ontologies are more limited in scope, being tuned to the task at hand, but allow rapid prototyping of ontologies to support novel applications. Researchers will initially produce multiple competing ontologies but in doing so will test their effectiveness.

Implementing an ontology

Scientists constantly create ontologies, but in the past they have rarely formalized them. It is really the advent of information sharing, and need for data integration from locations across the world, that has driven the growth in scientific ontologies. Data structures required by analytical tools are ontologies, including entity relationship diagrams or an object model used for databases. A structure may be implemented in many ways, depending on the applications that need to access it, such as a relational database schema or XML definition file (DTD, XSD etc.). Data exchanged formats are increasingly using XML. Each implementation will have its own instances e.g. an XML document using a particular schema. Even if you forego the explicit creation

of an ontology, often it's implicit in the implementations or instance data. Some implementations of Bioinformatics ontologies include MAGE-ML [26], GelML [27] and the XML represetation of the Systems Biology Ontology[54].

### 1.6.2 Databases

While omics experiments generate the high volume of data needed to characterize the structure and dynamics of biological networks, at the end of the experiment you are confronted with a large amount of raw data and a large amount of meta-data to manage. A single microarray (sample) will produce an intensity file with a size ranging from tens to hundreds of megabytes and thousands to millions of measurements; this problem is exacerbated with the newer technologies, where a single HTS run (lane) may produce files in the gigabyte range. This must be multiplied by the number of replicates, samples and conditions, resulting in total data size that is 10 - 100 times larger. The corresponding metadata (experimental and biological information describing the data), and the relationships can be extremely complex. The object model for the Minimal Information about A Microarray Experiment (MIAME, the MAGE-OM) contains 169 objects with very complicated relationships [56].

Relational databases have been the de facto standard for storing large amounts of data for two decades, and in the following sections, we'll discuss them and their limitations. Next, we'll briefly discuss NoSql (not-only SQL, non-relational and distributed databases), which have been gaining more prominence as the limits of relational databases have been pushed (and often exceeded) by the recent data explosion in both the sciences and the commercial sector. While NoSql databases are not the focus of this dissertation, the limitations they're meant to work around, and their general approach, apply to the Data-FATE system that we've developed, which is the subject of the research described in Chapter 2.

Relational databases

Relational databases have mature data management software support, and are used by nearly anyone needing to store and retrieve large amounts of data (e.g. corporations, researchers etc.). All relational databases adhere to the relational model (to varying degrees), which was developed by E.F. Codd (IBM) in 1970 [57]. The relational model itself is relatively simple to understand, but this simplicity comes with a cost: the software implementation that enforces the relational model is extremely complex and resource intensive. The data structures are manipulated and queried using the Structured Query Language (SQL), a standard honored at some level by all relational database management systems (RDBMSs). Although the language is relatively simple to learn and use, the actual algorithms for manipulating the relational data structures and retrieving the data sets are quite complex, and performance tuning is an art, not a science.

The relational data model

The relational data model imposes a particular type of logical view on the data structures, carrying with it an ontology. A particular instance of a relational model – such as one to model the operon structure of prokaryotic genes – is an implementation (realization) of an ontology, whose structure conforms to the rules of the relational model.

The relational model supports three types of general relationships between tables (objects): one-to-one, one-to-many and many-to-many. With enough creativity, one can use a combination of these relationships and metadata tables to create a wide variety of ontological relationships – hierarchical, part-of, is-a etc. – but the implementation can be tricky since context, semantics and meaning of the non-set relationaships are not inherent to the relational model. In fact, the available operators are quite limited and business rules must be used to enforce other types of relationships.

Relational databases for 'omics data

In addition to ontology development, bioinformaticians have spent the past decade developing core data models that provide a common structure to biological information, the Generic Model Organism Database (GMOD) being one such example [58]. The GMOD is based on the relational model, is extensible in defined ways that allow customization for unique aspects of an organisms biology, and has associated with it tools for populating , querying, visualizing and publishing the database instance [59, 60, 61].

Limitations of relational databases

Relational models do not encompass semantics, terms that shade interpretation by context and process. In addition they do not scale well in their standard configuration, a limitation that genomics labs are now hitting. There are parallel configurations [62, 63] used in the business world, but no freely available systems.

*Semantics*

The relational model cannot represent parallel valid relations between entities that are distinguished by a temporal function. They also handle many-to-many relationships by flattening them, using additional entities to bridge these relations. This means there can be many valid ways to faithfully represent ontological objects, attributes and relationships, leading to more proliferation of methods, and barriers to integration. There are also limits to computational clarity: columns have only a specific data type, but lack the ability to carry a semantic tag such as units, a feature of any quantitative measurement from a device.

*Processing*

Relational databases are limited by the hardware on which they reside. By design, these databases are meant to run on a single computer (server) and will have problems when one or all of the machine's resource thresholds are reached: disk space, disk I/O

speed, memory and processing power (CPUs). Even with a multi-terabyte RAID, a machine's disk space is quickly exeeded when storing large numbers of microarrays. Since relational databases retrieve information that is stored on disk, the disk I/O speed is also a major factor for improving the performance of queries. When running many types of queries, relational databases perform their operations in memory, which can quickly be exceeded by large 'omics data sets. Also, most servers have a hard limit on the number of processors that they can have, so even if you can convert your analysis routines to use many processors in parallel, you have a finite amount at your disposal. Finally, the standard way to store data of the same type is to load it all in a single table. This becomes problematic for large 'omics data sets, whether it be microarray data or HTS data, because of the indexing performed by the system to produce data addresses. To run SQL queries efficiently, you must therefore define secondary indices to speed them up. Creating indices takes a significant amount of time for large tables and the indices themselves require a significant amount of disk space. If creating an index were a one-time operation, this might be acceptable. However, when you generate more data and want to add it to your database, you will need to drop all of the indices, load the data and then re-create the indices. You can address some of these problems by scaling the server vertically – improving the hardware for the machine – but there is always a threshold for any single machine. You can also replicate your database in a so-called master/slave setup, but this only helps with handling large access loads, not the size of the database itself.

Given these limitations, developers have come up with some ingenious solutions. In some of them, developers still use relational databases, while in others, they move away from the relational model. In all of them, however, they use the same general guiding principle: partitioning the data (divide-and-conquer).

Sharding and NoSQL strategies

*Sharding* is the logical partitioning of rows of data, based on some shared characteristic (e.g. probe intensities from the same array), and storing them in separate tables, either on the same server or on different servers.

If you are constrained to one database server, you face hard limitations on disk space, memory and processing power. If you have multiple machines, you can alleviate many of these constraints because each server has its own database schema and operates independently from the others (a shared-nothing architecture). Most relational databases don't have any native support for automatic data partitioning or load balancing, so you need to write your own custom solutions.

Although custom software has been developed and marketed to ease the sharding problem, the products tend to be expensive and require significant maintenance. The difficulties and cost inherent in sharding have led many software firms to abandon relational databases and either develop, or use, non-relational solutions.

Non-relational databases don't adhere to the relational model and are usually called NoSQL (not only SQL) databases. We will use the more inclusive term non-relational database management systems (NRDBMS) to refer to them. . Distributed non-relational databases were specifically designed to solve the big-data problem: partitioning data across multiple machines and efficiently retrieving it. Data is retrieved through an application-specific API (using parallel processing), rather than a single, standard language such as SQL. Most of these systems do not index data as with relational databases, although some support it. Regardless, these systems store and retrieve data extremely fast. One major drawback of these systems is that they have varying degrees of consistency, as described by the CAP theorem [64]. Notable examples of non-relational databases are Bigtable [65], HBase [66, 67] and Cassandra [68, 69].

### 1.6.3 Workflows

Defining good data models (ontologies) and effectively storing, managing and retrieving omics data are critical concerns in data-heavy sciences, but these are only preliminary steps to the end goal of research: processing, integrating and analyzing the experimental data to characterize the system under study. Bioinformatics is overrun with analysis scripts, pipelines and data formats. Most computational experiments consist of many sequential analysis steps, each expecting different types of input data and producing different types of output data. To join each step, a glue (transformation) module (script) is required to transform the output format of one step into the format expected as input by the next step. This series of steps, tied together, is an analytical workflow; analagous to an experimental protocol in the wet lab. In Bioinformatics, these workflows quickly become large and complex, and, unlike their lab counterparts, are often poorly documented [70]. Managing these workflows becomes cumbersome quickly, reqiring the development of workflow management tools.

There are two major aims when developing tools to manage analytical workflows: tracking data provenance/data lineage and task automation. Data provenance refers to the documentation of your analysis - what exactly was done at each step. This is critical for science because all experiments – including computational experiments – must be reproducible. We also want to automate and abstract the workflow as much as possible. Analysis and data transformation steps can be viewed as independent modules that we can put together in multiple ways, depending on the experimental goals. We want to define each step in the workflow, connect them and hit run, without the need to oversee and initiate each step.

Workflow management systems aim to minimize human interaction during analysis and to make that analysis reproducible, which requires standardized analysis modules, data transformation modules and interfaces between them. These systems represent

a workflow as a graph, where a node is a specific analytical task and a directed edge between two nodes typically represents the flow of data from one node to another (i.e. the output of one node is the input of the other, via a specified communication channel and envelope). The systems then manage information about the server where a tool is located, and access and retrieval protocols. The most well-known and widely used Bioinformatics workflow management systems are Galaxy [71] and Taverna [72].

These systems suffer from one major limitation: the data interfaces – input and output – for an analytical task are not explicitly defined as an ontological type. Although many of the analysis modules support certain data exchange formats as input and output, they don't explicitly state what ontological type is expected or produced. Put another way, they're not semantically 'aware' of the data. Their purpose is really to automate processes over distributed services. Since an ontological type can be implemented in many equivalent ways, such as a database table, a delimited text file or a XML file, it would be easier to glue together modules if you knew the data types expected for input and produced as output. For example, a module might expect a 'gene expression' data type as input, which can be instantiated in various ways, and produces a list of differentially expression as output, which also can be instantiated in many ways.

### 1.6.4 Summary

The principle challenges when sharing any type of experimental data include finding mechanisms to communicate the meaning of the data and tools for disseminating the data so that it can be retrieved in useful parts as well as entire sets. Data models, database management systems, common vocabularies and shared workflows are all essential before a community can make use of a common resource.

An ontology allows us to more easily understand and organize data, allows for a consistent representation of data and gives domain experts a controlled vocabulary with which to communicate and standardize analyses. Ontologies are also essential for

communication, automated reasoning, and for integration of disparate data sources. Although having a prescriptive ontology is ideal, it needs wide community buy-in, a deeper understanding of a wide range of relations and stability. In an emerging domain, the interaction problems exists: representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the problem [73]. In many cases, a descriptive ontology simply happens, with or without the understanding that the relations are as important as the vocabulary definitions, because they still have many of the same benefits of a prescriptive ontology, excepting community buy-in and completeness. Although there will be multiple competing descriptive ontologies initially, the domain still benefits from having them, and they'll most likely be merged into a prescriptive ontology as the domain matures. Besides Data-FATE, which we've developed and will discuss later, we haven't identified any tools for building descriptive ontologies.

Databases allow you to more easily store, manage and retrieve data, as compared to simple file manipulation. Relational databases have been the de facto standard for years for any data intensive field, including Bioinformatics. However, the dramatic increase in 'omics data (and data in other domains) is straining single-server relational databases, causing many people to develop alternative approaches that use some type of data partitioning (sharding). Some have moved away from the relational model (so-called NoSQL databases), while others integrate partitioning methods on top of existing relational databases. In either case, none of the existing databases integrate ontologies into their systems.

To reproduce the results of a computational experiment, you need to document each process (data transformation, analysis script etc.) in the workflow - much like developing a protocol and keeping a lab notebook in the lab. Because there are numerous step to a computational experiments, and even more implementations of a particular type of analysis, workflow management systems have been developed to

keep track of and automate analysis pipelines. Several workflow management systems have been developed for Bioinformatics, and are a step in the right direction. However, none of them explicitly incorporate ontologies into the input and output interfaces between modules Doing so would make it easier to link separate modules into a single workflow.

In Chapter 2 (Data-FATE) we'll discuss how we've integrated ontologies, data partitioning and workflows into a single system.

## 1.7 Accuracy and reproducibility problems with microarray data

*** UNDER CONSTRUCTION ***

The expectations for microrrays were extremely high when the technology first appeared, especially for applying them to find multi-genic signatures for diseases (biomarkers). The hope was that physicians could use these signatures to screen for or diagnose a disease in very early stages, since it's always better to identify a disease before its symptoms appear. Requirements for screening are less stringent than for diagnosis, because any findings will be verified by an independent diagnostic tool, such as qPCR. However, researchers and doctors ultimately hoped to use microarrays for diagnosis, especially for complex diseases with many genes involved. If the molecular signature includes hundreds of genes, PCR validation is not feasible. To be used as a diagnostic tool, the FDA requires that a platform's coverage, accuracy, sensitivity, specificity, reproducibility be rigorously assessed and meet very high standards [74]. Although not an explicit requirement, this is imperative for systems biology research too. To understand complete regulatory networks we need to use a high-throughput, multi-locus experimental technology for studying systems (single-locus technologies are not feasible), but and missing and misleading data has a significant impact on characterizations of these networks [75] .

The requirements for interpreting microarray data essentially condense to four questions: How much of the target space does the experimental platform actually

measure? Does each probe measure the correct target? Does each probe measure a single target? How well correlated is the probe response with the target amount (i.e. how good is the binding affinity between the probe and target)? As discussed in an excellent review by Draghici [74], microarrays had problems with sensitivity, specificity, accuracy, and cross-platform reproducibility, and thus, did not live up to their early promise. This is largely because the technology was very complicated and changed so rapidly that researchers did not sufficiently examine these questions. Given these known reproducibility problems, what are the factors that cause them?

## 1.8 Factors affecting the response of microarray probes

Several factors influence the response of microarray probes, with some of them corresponding to meaningful biological variation while others are sources of unwanted variation. Although there is some overlap, the can break down these factors into four categories: experimental, biological, physical and technical. Of the four, only experimental factors are interesting parts of the experiment (signal) - the rest is noise.

### 1.8.1 Experimental and biological factors

*Experimental factors* are the variables that you control during a microarray experiment, that, when varied, produce the meaningful phenotypic changes in the biological system under study i.e. experimental factors are the sources of meaningful (desired) measurement variation. For example, the experimental factors could be disease state and the goal of the study might be to determine the difference in the transcriptomes of healthy and diseased tissues. *Biological factors* correspond to natural variation between individuals, such as differences in a genotypic population or differences between the "same" cells caused by stochastic processes. These factors are usually sources of unwanted variation, but can sometimes be controlled e.g. by using inbred lines in your experiment.

### 1.8.2 Physical and technical factors

Physical and technical factors (and composites of them) introduce unwanted variation into microarray measurements that have nothing to do with the experiment or biology of the system, but that still affect a probe's sensitivity, specificity, accuracy and precision. *Physical factors* directly affect the binding affinity between the probe and target, causing a bias in the amount of probe-target duplexes that can form, and in turn, biasing the scanned probe intensities; these factors correspond to the thermodynamic and biochemical characteristics of a probe and target. *Technical factors* correspond to variation introduced by microarray design, manufacturing, platform differences and experimental handling of sample material. *Composite factors* are a combination of one or more physical or technical factors. Here we'll discuss the factors relevant to this study: probe and target secondary structure, probe-to-target mapping, probe sequence motifs and sensitivity range of the scanner.

Probe and target secondary structure

Secondary structure –- present in the probe, target or both –- is a physical factor, directly affecting the binding affinity between the probe and its target. Although probe and target monomer structures sometimes increase the binding affinity (duplex stabilization), most often they have no effect or diminish the duplex stability. If the binding affinity between a probe and target is zero, the probe will indicate that the target is not present (false positive); if the binding affinity is non-zero but not perfect, the probe will underreport the amount of target present (negative bias).

This becomes complicated because a probe's sequence need not exactly match (be perfectly complementary to) the target's sequence to bind strongly [76, 77]. For example, the Kane criteria only requires a minimum nucleation of 15 nucleotides and 75% sequence identity between probe and target for hybridization [78]. Conversely, even if a probe sequence is an exact match to a target sequence, it's possible, due to thermodynamic and kinetic effects, that the binding affinity between them is zero.

For example, a probe or target might have internal secondary structure (monomer) that is more favorable than the duplex, a target homodimer and heterodimer might be more favorable than the duplex or the duplex might be kinetically unfavorable (although thermodynamically favorable).

If the concentration of probe and target were nearly equal, this would be a moot issue because a probe-target pair with an exact alignment will nearly always outcompete an inexact alignment. However, the probe concentration on all array types is significantly higher than the target's, in order to drive the monomer reactants to the duplex product. This allows less optimal, inexact alignments to bind to the same probe as well, without much competition.

Probe-to-target mapping

Probe-to-target mapping problems are composite factors that confound our interpretation of a probe measurement by making us think that a measurement reflects one target (the intended target), when it actually measures a different target or the intended target and a different target - this causes false positives and negatives. The most common mapping problems are cross-hybridization (to additional and unintended targets) and missing targets, as shown in Fig. 1.3. If a probe measures multiple targets, it's said to cross-hybridize, which typically causes false positives, false negatives and inflated estimates of target concentration; if a probe measures a single target for which it was not designed, it's said to have an unintended target, which results in both false positives and negatives; if a probe can't measure the target for which it was designed, it's said to have a missing target, which causes false negatives.

Probe-to-target mapping problems can happen for several reasons. Updates to the genome annotation are the most common, and widely understood, technical factor that can cause problems. Since probes – especially those on an expression or SNP/CNV array – are typically designed against exact alignments to a certain ver-

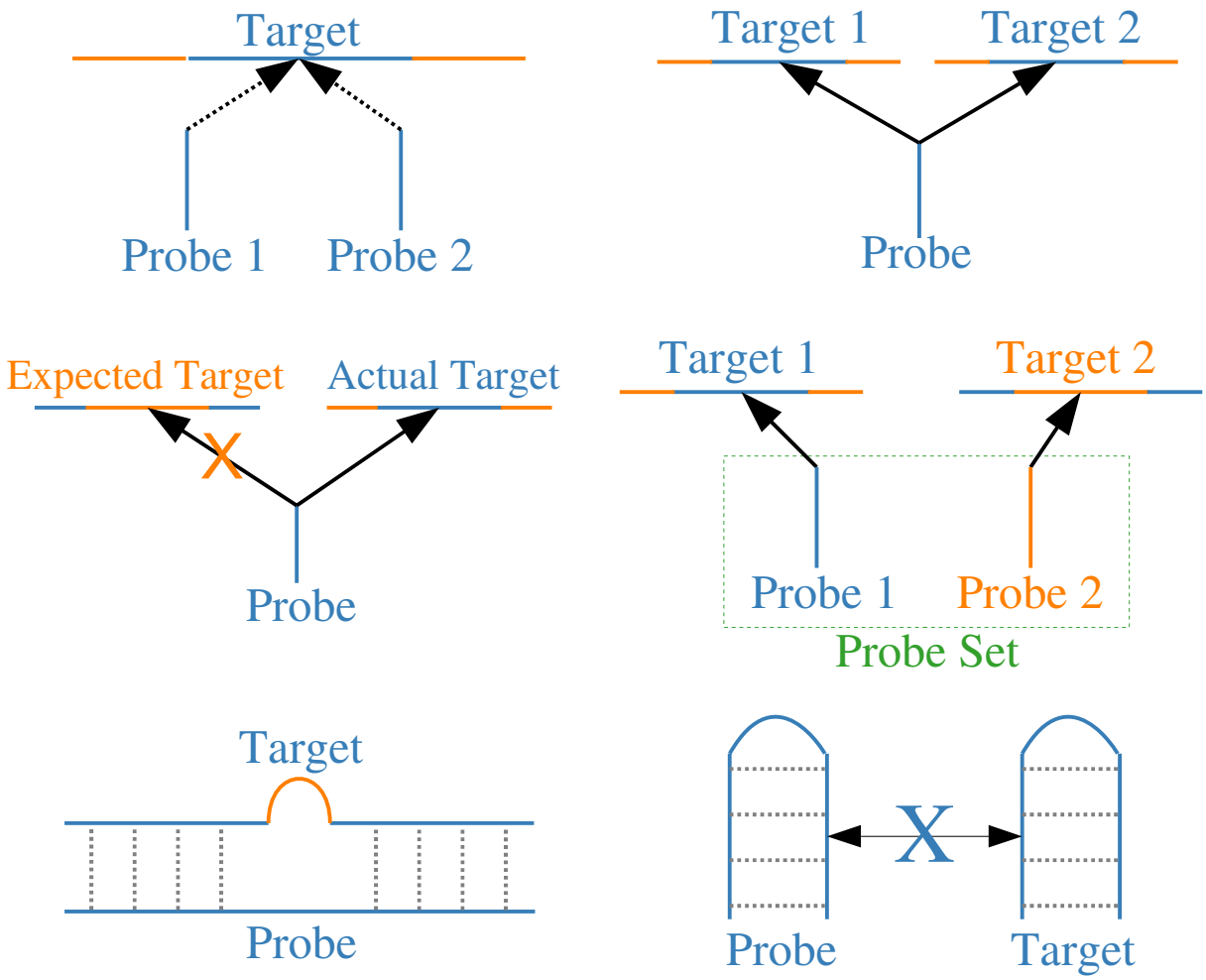


Figure 1.3: Common mapping problems.

sion of a genome annotation, any updates to the annotation will likely change the expected targets for a probe. Probes thought to have a single target with the previous genome annotation might now, with the updated genome annotation, be known to cross-hybridize, have a different target than intended or have no target.

High binding affinity between probes and targets with inexact alignments is a common, but not widely understood, physical factor that can cause probe-to-target mapping problems. Since none of the major microarray platforms designed their probes using inexact alignments to the genome, they most likely missed a significant number of potential targets for a probe. This means that a even if a probe has a single, correct target, based upon an exact alignment, it might still cross-hybridize to several other unintended targets to which it is not perfectly complementary. However, inexact alignments can also be an advantage. If we discover that a probe without an exact alignment has an inexact alignment to another target, we can re-purpose it to measure the other target, assuming the binding affinity between them is sufficiently high.

Probe sequence motifs

Probe sequence motifs, like secondary structure, are most likely a physical factor that affects the binding affinity between a probe and its target, although the mechanism is not always known. The two most common types are G-runs and primer spacers [79, 80]. Probes with a G-run motif have at least one instance of $\geq 4$ Gs in a row, while probes with a primer spacer motif have at least one instance of CCTCC. The latter is incorporation of a T7-binding site when a cDNA is amplified to cRNA during target preparation and is only a problem when the target is prepared in this manner; the problem arises because the probes were not designed against this feature and it causes cross-hybridization in some cases. Both sequence motifs result in probes that report a significantly greater intensity than the target amount should cause, and their expression profiles are more highly correlated with each other across

conditions, than they are with other probes designed to measure the same target (e.g. other members of an Affymetrix probe set). Whatever the mechanism, these probes usually overreport the amount of target present (positive bias).

Sensitivity range of the scanner (linear range)

For Affymetrix scanners, only reported intensities between 200 and 20,000 fluorescence units are consistent - this is the sensitvity range of the scanner [81, 82]. For genes that aren't expressed, the reported intensity is often below the sensitivity range ($< 200$), and behaves inconsistently. You can only say that the intensity, at most, was 200. For probes with a reported intensity above the sensitivity range ($> 20{,}000$), you can only say that the intensity is at least 20,000. Whether above or below, you must set and enforce limits on your interpretation of your measurement.

### 1.8.3 Summary

Several physical and technical factors introduce unwanted variation into a probe's measurements, confounding our ability to accurately interpret a microarray experiment. As with any measurement platform, the best policy is to identify affected sensors (probes, in this case) and then either fix or remove them.

## 1.9 Correcting for physical and technical factors

Both statistical and factor-based approaches have been developed to identify and adjust for probes whose intensities don't accurately reflect relative amount of the intended targets. Statistical approaches were developed first, followed by factor-based methods. Here we'll the merits and limitations of each approach.

### 1.9.1 Statistical methods

Statistical approaches, such as RMA, GC-RMA , dChip and MAS5.0 [83, 84, 85, 86], follow a common workflow: background correction, standardization, normalization, identification and removal of outlier probe measurements and, where necessary, summarization (particularly for Affymetrix arrays). There are many benefits to the statistical approaches: they're easy to use, good at removing generalized measure-

ment variation following an expected distribution and are already implemented and tested in major analysis packages, such as those found in BioConductor [87]. However, they handle the many factors causing variation by merging them into a single 'noise' factor. By not treating the factors separately (that is, knowing what is being and removed and why) meaningful experimental and biological measurements are likely to be removed along with the undesired technical and physical factors.

### 1.9.2 Factor-based methods

In contrast, factor-based approaches identify probes influenced by specific physical and technical factors and adjust the interpretation of the measurements accordingly. When a probe is affected by a factor, it's typically handled in three ways: it's deprecated, its relationship to a target is updated, or its reported intensity is adjusted (the second and third can both happen for the same probe). Several benefits recommend the factor-based approaches: you're less likely to remove the meaningful biological and experimental variation and the treatment of probes is consistent across all conditions. However, factor-based approaches require a significant amount of data management and can be computationally challenging, especially when creating usable files to reflect changes to the array specification. Despite these challenges, many studies show that, for any experiment, removing known factors that confound measurements improves the reliability and reproducibility of the data analysis.

### 1.9.3 Single-factor studies

Here we discuss several single-factor studies (relevant to our research) that have been performed on Affymetrix microarrays. We discuss the general approach and how, by accounting for these factors, the interpration of the experiment improved.

Probe-to-target mapping

Shortly after Affymetrix released the sequence information for their arrays into the public domain, several researchers analyzed the probe set definitions [88, 89, 90, 91, 92, 93, 80], identifying a number of potential problems with the original definitions that

could produce measurement error within a probe set. They then proposed several bioinformatics methods for re-defining the probe sets to solve these problems (e.g. creating a custom array specification), intending to reduce the measurement error and to make the aggregated measurements more biologically relevant. In many cases, these groups validated their re-definition strategies by showing that their custom probe set definitions, when compared to the Affymetrix default, significantly changed the differential expression results. In some cases, subsequent studies showed that the re-definition strategy significantly improved the correlation between microarray measurements and experimental results. The custom probe set definitions of Dai *et al.* [89], and two later studies using them [94, 95], illustrate how custom array specifications can significantly improve microarray measurements and the conclusions drawn from them.

For several Affymetrix expression arrays, Dai *et al.* [89] re-defined the original probe sets into gene-, transcript- and exon-specific probe sets. They used the most up-to-date versions of several public genome databases, such as UniGene [96] and Refseq [96], in this process, and then created custom CDFs for each source. In one case, they used an updated version of UniGene to define a gene-specific CDF for the Affymetrix HG-U133A chip and then reanalyzed data from a cardiac tissue study (GSE974) [97]; comparing the updated CDF and the original CDF, they found between 30-40% differences in those genes predicted to be significantly differentially expressed between the two. When performing a similar analysis with other custom CDFs, they found between 30-50% differences in predicted differential expression. Subsequently, Sandberg *et al.* [95] showed that Dai's custom probe set definitions, when compared to the original definitions, improved the accuracy and precision of transcript estimates for a set of cross-lab replicate arrays [98]. In particular, their accuracy metrics showed that the microarray measurements became more similar to those measured by RT-PCR. Later, Mieczkowski *et al.* [94] showed that Dai's

custom CDFs significantly improved the correlation between microarray expression profiles and RT-PCR expression profiles. Thus, re-defining array specifications can potentially improve the down-stream analysis of Affymetrix microarrays.

Sequence motifs

Upton *et al.* [79, 80] reported that probes with certain sequence motifs have intensities that are uncorrelated with the other probes in the same probe set; however, they tend to correlate well with any probes having the same sequence motif, regardless of probe set membership. In this case study, we will focus on the two major types of problematic sequence motifs identified by Upton *et al.* [80]: G-runs and primer spacers. Probes with the G-run motif, $\geq 4$ Gs in a row, tend to produce consistently high intensities, with some position dependence. The primer spacer motif, CCTCC, is related to the incorporation of a T7-binding site when a mRNA is amplified during target preparation. When the target is amplified in this manner, the probe intensities tend to be higher than most, introducing a spurious correlation similar to that seen with G-runs. Since both of these sequence motifs introduce a systematic bias when summarizing probe set intensities, any probes including them should be removed from a CDF prior to calculating expression values. This is always true for the G-run motif and is true for the primer spacer motif when the target is amplified by incorporating a T7-binding site.

### 1.9.4 Multi-factor studies

While several studies have investigated and corrected for individual physical and technical factors, few studies have integrated them into a consolidated pipeline. Here we discuss two such studies, BaFL and Lo-BaFL. The BaFL pipeline is the basis for the research in Chapter 4.

Biologically-applied filter levels (BaFL)

Thompson *et al.* [99] developed a "white box" pipeline – Biologically-applied Filter Levels (BaFL) – to identify and filter microarray probes that are likely to report in-

correct or misleading intensities based upon certain biological properties, such as the presence of SNPs in the probe sequence (e.g. as identified by AffyMapsDetector [100]), probe cross-hybridization, internal structure in either the probe or target sequence that reduces binding affinity, and probe intensities that fall outside the sensitivity range of the scanning device **[81, 82]**. They tested their filter set on two independent microarray studies of human lung adenocarcinoma and showed that it improved concordance between lists of significantly differentially expressed genes and sample classification.

Lo-BaFL

Baciu et al. [ms in review] developed the Lo-BaFL pipeline, which adapts the orignal BaFL pipeline to the Agilent platform and also extends it by accounting for the Upton sequence motifs.

### 1.9.5 Summary

Factor-based methods identify and correct for probes influenced by known physical and technical factors. Although purely statistical methods are easier and less computationally intensive, factor-based methods are preferred because they're less likely to remove meaningful sources of measurement variation. Many single-factor methods have been developed for dealing with probes affected by incorrect probe-to-target mapping, interfering secondary structure, problematic sequence motifs and scanner limitations. However, by only accounting for a single factor, your interpretation of the experiment, while better, are still suspect. To comprehensively improve an array, you need to account for all factors known to affect probe measurements. Two such multi-factor pipelines have been developed, BaFL and Lo-BaFL (an extension of BaFL to the Agilent platform), and have been shown to dramatically improve experimental interpretations. To date, a version of the BaFL pipeline doesn't exist for the Affymetrix E. coli arrays.

To evaluate the impact of a filter or filter set, you need to compare your measure-

ments to a standard or make relative comparisons. All of the factor-based approaches evaluate the impact of their changes using these approaches, but they only use a small number of the available options (one or two), limiting the evaulation of the true impact. Do your changes improve some evaluation methods, but not others? Why? By using many evaluation methods to assess the impact of your changes, you lend further confidence to and support for your changes. Also, it has another benefit - any major discrepancies between the evaluation methods suggests that another, unknown factor could be present. Have we accounted for all of the factors? Diagnosing the array in this way is an important step, and none of the existing factor-based methods address this problem. We'll discuss this more in Chapter 4 (ProbeSieve).

## 1.10 Custom specifications for factor-based methods

We've discussed several factor-based methods to correct for unwanted variation introduced by physical and technical factors. To implement these corrections, you need to either modify the downstream analysis routines to account for them, or you need to modify the logical design contained in an array's specification file. In almost all cases, updating the specification is simpler and less error-prone than modifying analysis routines. However, understanding the logical rules and file formats for an array's specification can still be quite challenging.

Here we first discuss the Affymetrix Chip Definition File (CDF) format used to describe the logical design of all of their arrays, the specific logical rules for constructing probe sets for Affymetrix 3' expression arrays, and how these rules are instantiated as a CDF. We then discuss methods and tools you can currently use to create and communicate custom array specifications, emphasizing the benefits and limitations of each. We conclude by discussing what is needed to simplify the process of creating a custom CDF, which is the background necessary for the Chapter 3 (ArrayInitiative).

### 1.10.1 Physical vs. logical design of Affymetrix arrays

The *physical design* of an Affymetrix microarray corresponds to the actual probes on the array. The *logical design* of an array corresponds to the set of probes that you intend to use, and any higher level groupings of them that you intend to use. For example, Affymetrix arrays have multiple, affiliated probes grouped into a probe set, designed to measure different locations on the same target. A probe set is part of the logical design of the microarray. The probe pairs defined for expression arrays are also part of the logical design.

The physical design of any array is fixed: once the probes are synthesized onto the array, you can't change them. However, you can change the logical design of the array by deprecating probes or re-assigning probes to different probe sets (note, this is specific to Affymetrix arrays although a probe can be assigned to a different gene on other platforms, however this will leave the original target without a measurement). For arrays with only one probe per target, deprecating many problematic probes might dramatically reduce the number of genes that you can monitor, possibly compromising the types of analyses you can perform or make it impossible to monitor a specific subset of targets (e.g. genes in a pathway). Having multiple affiliated probes in a probe set makes Affymetrix arrays more robust to changes in the logical design.

Chip Definition File (CDF)

Affymetrix uses the Chip Definition File (CDF) format to provide metadata about the array and to specify its layout e.g. location of probes, logical grouping of probes into probe sets, etc.. This format is used for all of its arrays, although there are two different major versions (ASCII, a text-based format, and XDA, a binary format), multiple minor versions of each. The probe set is the central logical grouping of probes and are defined as a hierarchical combination of units, unit blocks and cells. A CDF can have multiple units, a unit can have multiple unit blocks and each unit block can have multiple cells. Using this hierchical structure, Affymetrix defines multiple

types of probe set: expression, genotyping, copy number, custom sequence or tag. The type of probe probe set used depends upon the array type. For example, a 3' IVT expression array only contains expression probe sets. Each unit corresponds to a probe set, a unit can only have a single block and the block contains multiple cells, which correspond to the probes. The format of a CDF and the specific logical arrangement of units, blocks and cells for different types of probe sets make creating a CDF a rather complex and daunting task.

### 1.10.2 Communicating and using a custom logical design

Although several researchers developed custom logical design for Affymetrix arrays (discussed below), they communicated their re-definition strategies in a variety of ways. Some of them published their *general strategies* for re-defining the probe sets, without providing custom specifications for individual microarrays; others published custom array specifications for a limited subset of microarrays, although in a file format different from the standard CDF format; still others provided custom CDFs, but again, for a limited subset of microarrays. For those research groups who can simply use a provided custom CDF, this bewildering variety of formats does not pose a problem. However, it is a problem for those groups who are not in this fortunate situation: those who want to use a published re-definition strategy, but don't have access to a custom array specification file (non-standard or standard); those who want to modify an existing method; those who want to combine multiple re-definition strategies; and those who want to develop and implement their own re-definition methods. For example, one of our research interests is to test different gene models by assigning probes to transcript-specific sets and then creating model-specific CDFs. What are the options for these researchers?

One option is to create custom versions of the algorithms for summarizing probe set intensities, such as RMA. However, writing these custom algorithms is likely to be daunting, error-prone, and hard to test. A better option is to create a custom CDF.

Researchers can then generate summarized probe set intensities using any of the well-accepted and tested analysis packages provided by Affymetrix or Bioconductor[87].

### 1.10.3 Current options for creating a custom CDF

Creating a custom CDF, though easier than modifying analysis algorithms, still presents several challenges. In the worst case, creating a custom CDF from scratch, researchers need to thoroughly understand the file formats (ASCII, XDA) and platform-specific logical rules for defining probe sets (3' expression arrays vs. exon arrays vs. SNP arrays) necessary to parse and write CDFs — a daunting and error-prone task. Using an existing application programming interface (API) or software development kit (SDK), such as Affymetrix's Fusion SDK [101] or *affxparser* [102] (an R wrapper of the Fusion SDK), is an easier and more efficient solution than writing in-house methods for reading and writing CDFs. However, this still requires a degree of proficiency in a specific programming language (C or Java for the Fusion SDK, R for *affxparser*), knowledge of the CDF file formats and probe set construction rules, and knowledge of the language-specific data structures for representing a CDF.

### 1.10.4 Summary

Many factor-based methods have been developed to improve our interpretation of microarray experiments, and the results of these studies have been reported in numerous ways. If you want to use a modified specification, the best approach is to create a custom CDF, which is the standard format. However, some researchers only published their general strategy for changing the specification, some published the changes in a non-standard format, while others did provide a custom CDF. Regardless, you won't always have access to a custom CDF for your array, and if not, you'll need to create this. Since both the CDF format and logical design of probe sets is complex, this is a daunting and error-prone task. Some APIs and SDKs exist to make it easier, by they still require knowledge of the CDF file formats and probe set construction rules. To aid those researchers who are not as computionally savvy, or simply don't

want to learn the formats and rules, a tool that only requires the most basic knowledge is needed, and is the subject of Chapter 3 (ArrayInitiative).

## 1.11 Dissertation outline

We performed three studies to address several of the open problems in handling large data sets and interpreting the results from expression experiments using Affymetrix microarrays. In Chapter 2, we introduce the Data-FATE framework (develoed by Carr and Weller), which integrates ontologies, data partitioning and analytical workflows, to improve efficiency and consistency when handling scientific data sets, with an emphasis on microarray data. Data-FATE is the only system that lets you build descriptive ontolgoies, which is accomplished by extending the relational model and its corresponding relational database management system. After introducing the key ontological concepts, we discuss our significant re-design of the Data-FATE Scientific Information Management System, including conversion to a standard three-tier architecture, development of an object-oriented Ontological Model API, a comprehensive re-design of the GUI to improve navigation and to make it more similar to GUIs for relational databases, a new set of tools to improve definition of ontological data types and for importing data, and several other features. In the next two studies, we develop methods and tools that help us to examine the impact of physical and technical factors known or suspected to dramatically alter the interpretation of a microarray experiment.

In Chapter 3, we introduce ArrayInitiative, a tool that simplifies the process of creating custom CDFs. We developed it so that we could easily re-design the array specifications for Affymetrix 3' IVT expression arrays, which is essential for testing the impact of the various factors (as we do in Chapter 4), and for making the framework easy to communicate and re-use. To test its usability, we also developed a case study to examine the impact of standard array filters on the interpretation of an array's measurements. ArrayInitiative was published in BMC Bioinformatics, 2011

(http://www.biomedcentral.com/1471-2105/12/136/).

In Chapter 4, we systematically examine the effect of physical and technical factors – both generally accepted and novel – on our interpretation of dozens of experiments using hundreds of *E. coli* Asv2 microarray. We applied and extended the BaFL pipeline for the Asv2 array and identified probes affected by one or more, such as probe-to-target mismapping, interfering secondary structure and sequence motifs, to name a few. We then definied several new specifications, both filtered an unfiltered, with probe sets designed to report on genes, transcription units and operons, and then created a corresponding CDF for each one using ArrayInitiative. We evaluated then impact of the specification changes by developing and applying an integrated set of novel and published evaluation methods. Of note, we introduce the concept of a response group, which is useful for evaluating the impact of a filter and for determining if there are hidden factors not accounted for by the filter set. To the best of our knowledge, no other such method exists.

# CHAPTER 2: DATA-FATE

## 2.1 Introduction

The volume and complexity of 'omics data is dramatically increasing, exacerbating the challenges already associated with experimental data: modeling the data (including the development of ontologies); storing, managing and retrieving the data (databases); analyzing the data (analytical workflows). These challenges demand that we develop better and more efficient methods for using 'omics data.

In the sciences, and especially Biology, we develop ontologies to model the complex entities and relationships present in our data and to provide a controlled vocabulary. Whether this is explicit or implicit, we always impose a structure on our data when analyzing it. As discussed in Chapter 1, ontologies are either prescriptive or descriptive. A mature scientific domain or experimental platform is likely to have a prescriptive ontology, which is a top-down, established set of entities and their relationships. While these provide a stable and consistent representation of data, they tend to be quite large, cumbersome, fragile or resistant to change. The taxnomic tree of life is an example prescriptive ontology, which imposes a hierarchical model of descent on the data; it breaks when you try to add non-hierarchical mechanisms of descent, such as horizontal gene transfer. In a less mature scientific domain, we tend to develop descriptive ontologies which are inferred from the data. While these allow for a consistent representation of data within a small group of researchers, they tend to change frequently and be different between groups e.g. gene names vary between groups studying the same organism and across organisms. In summary, a prescriptive ontology is ideal for a mature scientific domain; a descriptive ontology, for a

frontier or niche domain. Much of genomics research is concerned with the discovery of unknown cellular components. , In the absence of standardized tools to create descriptive ontologies when working with data, there is a need to develop methods and tools to automatically document the structure, so that we can record and share these ontologies.

The nature of the process dictates some of the requirements of an effective system. Data management systems must be able to store large amounts of data, efficiently retrieve specified subsets of it and allow researchers to create meaningful data structures (providing consistent representations and seamlessly coping with changes to the data model). Relational databases are the most widely used data management system, within and outside the life sciences. Although there is a growing movement away from the relational model when dealing with the very large high-throughput sequencing (HTS) data sets (e.g. Bigtable [65], Hbase [66], Cassandra [68]), when analyzing miroarrays, the focus of this dissertation, relational databases are largely sufficient. They do suffer from two serious limitations: first, it's difficult to represent many types of relationships used ontological models with a relational database; second, relational databases are server-bound for scalability so efficient use of memory is important. Given these limitations, there's a need to develop and implement methods for integrating ontologies within the relational model and to develop methods to optimize the storage and retrieval of desired data sets in a relational database. Many relational databases claim to integrate ontologies, but they do this by linking out to an external system, not by actually making use of the structure to reason within the immediate framework.

Data processing pipelines are also quite complicated for most types of genomics data. Although such workflows would seem to be a natural fit for the development of an ontology, the current state of practice is to separately employ and store configurable pipelines. As the data formats become more complicated and the number of distinct

analysis steps increases, stitching them together into a coherent and reproducible sequence of steps is becoming ever more challenging. In response, Bioinformatics researchers have developed several workflow management applications, such as Galaxy [71] and Taverna [72], to make the process easier. All of these applications have two main goals: maintain a record of data provenance/lineage and facilitate automation of workflow steps. Most provide a graphical interface that allows you to define a series of analysis and data transformation steps from those present in the library. Once defined, you upload your dataset as a file, hit 'Run' and the application runs them in the sequence indicated and delivers the output file. However, these systems suffer from one major limitation: the data interfaces – input and output – for an analytical task are usually not explicitly defined, and even if they are, they're often not used consistently. Also in most cases you cannot select a set of parameters for each module of the pipeline, or if you do this cannot be saved as semantic meta-data. This is because few standards exist to guide data providers.

To address these issues, Carr and Weller developed the DataFATE framework [103], which provides two novel extensions to the relational model: the ontological model and the scientific information management system (SIMS). The ontological model extends the relational model by integrating it with ontologies. The SIMS extends the relational database management system (RDBMS), providing a suite of tools that implement and manage the ontological model. It was designed for small teams, not for use by large communities over the Web. The DataFATE framework provides more consistent representations of experimental data, formalizes descriptive ontologies and improves storage and retrieval of large data sets by sharding data. Carr and Weller developed a prototype SIMS to test the framework, which confirmed certain expectations and generated a much larger set of new requirements for the SIMS.

In this chapter, we describe the steps taken to comprehensively re-design the

DataFATE SIMS, based upon the identified limitations of the previous version. The goal was to create a tool that would support network research by allowing you to easily manage thousands of microarray experiments and adding new layers of hierarchy to probesets as ontological definitions. First, we converted it to a standard three-tier architecture: graphical user interface (presentation tier), Ontological Model API (OM-API, logic tier) and database (data tier). In addition to consolidating the logical or business code into a single API, which can now be used by any third-party application, we also re-designed to be object-oriented. We then re-designed the graphical user interface (GUI) so that it was more similar to the GUI for a relational database, which significantly improved navigation, viewing of ontological data types and viewing of imported data and associated metadata. We also added several new tools to improve offline definition and bulk import of ontological data types and data. Finally, we added support for creating and managing multiple databases.

## 2.2 Data-FATE framework

The Data-FATE framework is both a data model and a management system, similar to the relational model and its corresponding relational database management system. The *ontological data model* extends the relational model, while the *scientific information management system* (SIMS) sits on top of the RDBMS (higher level abstraction) and implements the ontological model.

### 2.2.1 Ontological data model

The ontological data model extends the relational model by defining two novel data elements, *quantitation types* and *quantitation type sets*. These new data elements integrete ontological concepts to the relational model, providing an additional layer of abstraction. Both of the core ontological data types provide context and meaning for the values stored in the relational database. It allows you to unambiguously answer questions like "What kind of data does this table store?" and "What type of data is stored in this column?" – which is different than specifying simply that this column is

a integer i.e. semantics over syntax – and to define bottom-up, descriptive ontologies for your data.

#### 2.2.1.1 Quantitation type

A *quantitation type* (QT) defines a single quantitative or categorical data type, such as fluorescent intensity or genomic strand, serving as a bridge between an ontology and the relational model. It corresponds directly to an attribute in an ontology and a column in a relational database (e.g. a float, integer or character column).

#### 2.2.1.2 Quantitation type set

A *quantitation type set* (QT set)is a logical grouping of QTs into a set. It corresponds directly to a concept in an ontology and a table in a relational database, giving context to groups of quantitative and categorical data. In addition, the QT set, when considered as a single object, is a composite quantitation type.

#### 2.2.1.3 Relationships between quantitation type sets

The relationship between QT sets does not have an ontological/contextual meaning. A relationship between sets has the same meaning as it does in the relational model: a relationship exists, as defined by the foreign key, but there is not any context/meaning to it. Since a QT set is also a quantitation type, you can create ontological hierarchies.

#### 2.2.1.4 Example and implications

Once you define the quantitation type sets for the data in an experiment, you've defined the descriptive ontology for that experiment. For example, if you define a quantitation type set for the data in an Affymetrix CDF, you now have a standardized representation of this data type. The combination of quantitation type sets in an experiment represent a working descriptive ontology for a particular Bioinformatics domain, such as analyzing Affymetrix microarray data. Since a QT set is unambiguously defined, it can also be used to define explicit (standard) interfaces for analytical modules.

### 2.2.2 Scientific information management system

A *scientific information management system* (SIMS) is an extension of the relational database management system. Just as a RDBMS implements and manages the relational data model, a SIMS implements and manages the ontological data model. When developing a SIMS, two options are possible: develop it from scratch or build it on top of an existing RDBMS. The former option allows you to directly integrate the ontological data model into the system, but it requires a substantial amount of development and expertise with RDBMSs; the latter option allows you to leverage all of the features already implemented in an existing system (e.g. a standardized query language (SQL), query optimization, referential integrity, management of the physical data structures), signficantly reducing the development effort. However, this might make it difficult to implement certain features of the ontological model. We favor building a SIMS on top of an existing RDBMS, especially as a proof-of-concept.

Regardless of the approach, a mature SIMS should have the following features: (1) all of the features of a RDBMS; (2) an ontology query language; (3) tools for managing the core ontological data types (hidden from user); (4) tools for visualizing ontologies; (5) tools for defining ontological relationships; (6) query speeds on par with a RDBMS; (7) tools for workflow definition/tracking/visualization. Currently, the Data-FATE SIMS supports features 1, 3, 6 and 7.

### 2.2.3 Advantages and disadvantages

The Data-FATE framework (ontological data model and SIMS) offers several advantages to researchers with data-intensive experiments and analyses. The framework allows you to select the minimal set of data for your experiment. There are usually many attributes in data files from experimental platforms that are either poorly understood or completely unnecessary, which wastes disk space and complicates a table. The parts of the data that you use are determined by the QTs and QTSets that you must define before importing data, which forces you to understand the data first and

analyze it second. This reinforces the proper interpretation of your data – types of data (QTs) are expressed in terms of units and sets, not simply as relational data types. This also allows you to develop a specific ontology more comprehensively and explicitly than can be achieved by defining the tables in the relational model. By defining QTs and QTSets, you guarantee that the system can handle large amounts of data consistently. For example, once you define a QT Set for data files produced during an Affymetrix microarray experiment, whether probe design or target read-out (respectively CDF or CEL files), the Data-FATE framework guarantees that future data imported using the same QT Set will have exactly the same definition (or data type). When data types are unambiguously known, it's easier to automatically check, merge and create workflows. For example, you can automatically merge different data sets using their common fields (defined by the QTSet) and you can define the input and output interfaces for different steps in a workflow using QTSets. The SIMS creates partitions of the data, which means that it creates a new table for each data set imported (described below). This has two benefits: it improves the performance of database operations, such as indexing and retrieving data, and it allows you to extend the definition of a QTSet without needing to modify any existing tables.

While the Data-FATE framework offers significant advantages for the management, storage and modeling of omics data, developing a mature system is properly the job of a team over the course of several years. Developing a complete SIMS – with full ontology management, an extension of the SQL language, ontology visualization etc. – is nearly as complex and difficult as developing the underlying RDBMS; that is to say, a substantial development task. Also, a consequence of facilitating creation of descriptive ontologies is that in the short term there will be a proliferation of ontological terms and relationships. Ideally these will be compared, merged and refined, but this requires its own set of tools, development that is outside the declared framework of the Data-FATE project.

Described below are the prototypes that we implemented for a subset of the features necessary for the final SIMS. In the next sections, we discuss the two most recent prototypes of the Data-FATE SIMS.

## 2.3 Previous version of the Data-FATE SIMS (1.4)

The first prototype version of the SIMS (1.4) was developed by Carr and Weller to test the use and expectations of ontological model. Here we detail the features of this version of the SIMS and discuss its limitations, which directly informed the requirements for the next prototype version, 1.4.5, discussed in the next section.

### 2.3.1 Implementation

We developed the previous version of the Data-FATE SIMS as a two-tier application, consisting of a client application and a relational database backend. The client application was implemented as a desktop GUI using PyQT [104], a Python [105] binding of QT [106] from Riverbank Computing. We used Postgresql [104] as the backend database. Each of these third-party components are cross-platform and freely available.

### 2.3.2 Account administration

When first using Data-FATE, you are prompted (and required) to create a user account. Although Data-FATE is intended to be a single user system, after creating the first user, you can add new or existing users to your database.

### 2.3.3 Experiments

After creating a user account, the first step is to create an experiment. You always work within the framework of an experiment, which is a logical collection of data sets supporting a single research task, or project. Although only one experiment is required, multiple experiments per Data-FATE instance are allowed. When creating an experiment, one option is to associate QT sets with it (described below). The SIMS provides tools to create new experiments, update an existing experiment and associate QT sets with an experiment.

### 2.3.4 Curating ontological data types

Before uploading a data set, both quantitation types and a quantitation type set must be defined (if not accepting the default set). The SIMS provides tools to define new quantitation types, define a new quantitation type set, add QTs to an existing QT set and remove QTs from an existing QT set.

Defining a QT

When defining a QT, you must specify a name for the QT, an experiment type, a quantitation set type and a relational data type. Optionally, you can specify a unit (e.g. Joules) and a description. In addition, the tool lists all of the existing QTs, which can be filtered, so that you can check if the desired QT already exists.

Defining a QT set

A QT set can be defined once one or more QTs has been defined. This requires a name, an experiment type, a QT set type and one or more QTs. Optionally, one or more QTs may be declared the primary key. To create a new QT set that is similar to one that exists, you can copy it, modify its definition and save it with a different name. Re-defining a Q

Re-defining a QT set

You can re-define an existing QT set, with some restrictions. If the QT set is not associated with any data set, QTs may be added or removed from its definition. If the QT set is associated with a data, you may force- add a QT to the QT set, but you can't remove a QT from the QT set.

Pre-loaded ontological data types

The SIMS is pre-loaded with QTs and QT sets associated with microarray experiments. Specifically, it provides QT sets for Affymetrix CDFs, Affymetrix CEL files, Agilent arrays and QuantArrays.

### 2.3.5 Importing and exporting data

The SIMS has tools for loading 'generic' data and Affymetrix-formatted data. The generic data loader is designed to load any delimited text file; the specialized Affymetrix data loader is designed to load CDF and CEL files by first converting these custom formats into a delimited format. Loading a delimited file is the common point for the two tools. After specifying the input file, the data loaders create a unique name for the output table, although you can create a specific name prior to loading. Subsequently you associate a QT set (defined in advance) with the data set. Finally, the QTs in the QT set are mapped to the columns in the data file. Every field in the data file does not require a QT, but each QT must be mapped to a field in the data file. For example, the data file may have 20 columns while the QT set only has 5 QTs. You must map each of the 5 QTs to a field in the data file, which the SIMS will load while ignoring the rest. The order of fields in the data file doesn't matter. The SIMS will rearrange the data as necessary when loading and will also remember this mapping for future uploads.

After associating a QT set and mapping columns to a data file, the SIMS dynamically allocates (creates) the table and loads the data. These tables are not part of the core system. The SIMS creates a new table for each loaded data set, even for those that are associated with the same QT set. We decided to horizontally partition (shard) the data sets to improve performance. Since this is intended to be a single user, single machine system that extends the relational model, distributed non-relational databases were not an option; horizontal partioning is the most valid method for partitioning data in this scenario, creating multiple small tables, thus improving performance when adding, updating and query data. GMOD [58] also takes a small table approach, but the tables are highly normalized and it is constrained to use community-generated, top-down (prescriptive) ontologies.

### 2.3.6 Querying data

To query data, you submit queries directly to Postgresql, either using the command-line interface or a GUI, such as pgAdmin.

### 2.3.7 Limitations

This version of the Data-FATE SIMS was designed to be a prototype, quickly developed for testing with analysis use cases so that the cost of redesign would not be prohibitive. Here we will discuss the major limitations, which served as initial requirements for version 1.4.5, discussed in the next section.

#### Navigation

The original version of the SIMS lacked a centralized, global view of the core data types and other SIMS objects. For example, it requires inspection of multiple forms in order to determine the what experiments have been defined, the QT sets associated with them, the definition of a QT and QT set and other important information.

#### Curating ontological data types

This version provides tools to define ontological data types, QTs and QT sets. While you can re-define a QT set, there's not an analagous tool to re-define a QT. In addition, you must define QTs and QT sets individually, through the interface, which can be tedious. Also, some of the functionality which could reside in a single tool is split across multiple forms.

#### Importing data

The available tools to load data, both generic and Affymetrix-specific, required that data tables be loaded one at a time, even though the researcher has already defined the experiment set.

#### Architecture and application code

In this version, the business logic is embedded in the GUI code, so that code is duplicated across several tools. Consolidating this code into a centralized API (middle tier) conforms to standard practice, reducing duplication and possible errors.

Number of databases

This version of the SIMS only allows you to create one database per instance of the Data-FATE SIMS, although you can create multiple experiments. This is unncessarily limiting from both a conceptual and implementation perspective. Conceptually, a single study might have multiple experiments. When importing and organizing your experiments in Data-FATE, it might make sense to partition each study as a separate database, with each database having multiple experiments. From an implementation perspective, since all RDBMSs support the creating of multiple databases, it makes sense to provide the same support in the SIMS.

## 2.4 Current version of the Data-FATE SIMS (1.4.5)

Testing of the prototype version of the SIMS (1.4) confirmed certain expectations and generated a large set of requirements for the next version. Here we discuss those new requirements, and the new and updated features of Data-FATE SIMS introduced in version 1.4.5 to address them. The general requirements for this version were as follows: (1) add support for multiple databases and the ability to automatically create them; (2) change to a three-tier architecture with an object-oriented design; (3) improve navigation of the user interface; (4) streamline the definition of ontological data types; (6) streamline the loading of data sets; (5) workflow tracking.

### 2.4.1 Databases

The previous version of the SIMS only supported a single database, although it could have multiple experiments. In this version, users can create and manage multiple databases and easily switch between them. As before, each database can have multiple experiments. The SIMS now automatically creates databases, rather than manually running a script, as you needed to do in the previous version.

### 2.4.2 Three-tier architecture

The previous version of the SIMS used a two-tier architecture, consisting of a user interface (client) and a Postgresql backend. With this design, the business logic is

usually integrated with the interface and/or database code. This was effective for a prototype that changed quickly, but is not ideal for a mature system for two reasons. First, without centralized code, the business logic is often duplicated in several places, meaning that you have to update the same code in multiple places. This required more effort for development and maintenance of the code and greatly increases the chance to introduce functional inconsistencies. Second, a change in the interface or backend requires that you also rewrite the business logic.

In light of this, we changed the SIMS to a standard three-tier architecture: user interface (client), middle tier (business logic) and a Postgresql backend. Consolidating the business logic into a single API minimized redundancy and the errors associated with them. In addition, we can now distribute the API independent of the user interface if someone wishes to use the ontological model, while using their own Python scripts or user interface. Researchers are not constrained to use our adminstration tool. In addition to consolidating the business logic into a single API, we re-developed the code using an object-oriented model, discussed below.

### 2.4.3 Object-oriented design

We designed the middle tier using an object model, centered around the core data types (QTs and QT sets) and the Data Table instance type. We adopted an object-oriented design as being more maintanable, reusable and scalable.

### 2.4.4 Navigation

We improved navigation by developing a hierarchical browser which displays databases and their associated experiments, QT sets associated with each experiment, QTs that are part of each QT set and data tables associated with each QT set. This provoides a simple view of all of the main objects and their relationships, plus context-sensitive access to the main menu items associated with that particular object. We also added a dashboard which displays a summary of a selected item. For example, when you select a QT, the dashboard displays the definition of the QT plus the number of asso-

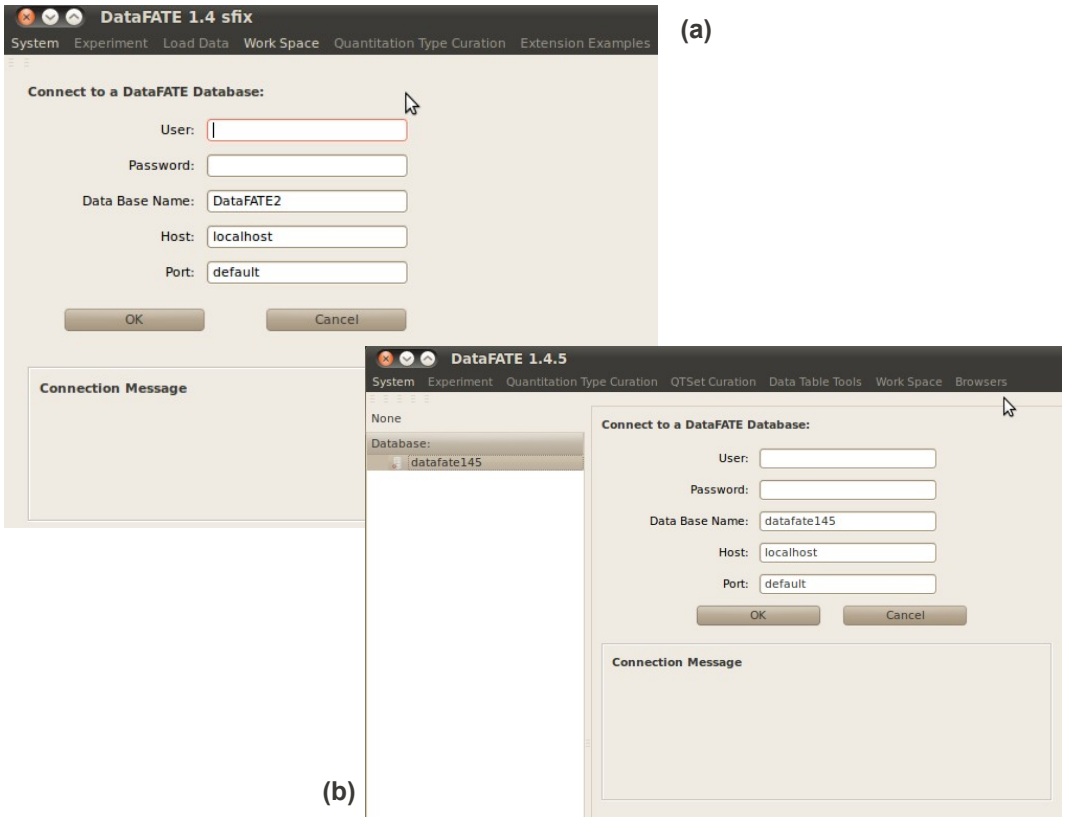


Figure 2.1: Comparison of Data-FATE logon screens. (a) The logon screen of version 1.4. (b) The logon screen of version 1.4.5, which includes support for multiple databases, as seen in the left navigation bar.

ciated QT sets and Data Tables. These changes to the interface, shown in Figures 2.1, 2.2, 2.3, make the navigation and display of information more similar to the interfaces for a RDBMS.

### 2.4.5 Curating ontological data types

All of the tools for curating ontological data types in the previous version are included in this one, but we re-organized the main menu structure to include separate sections for QT curation and QT set curation, making it easier to find the desired functionality. We also added three new tools: (1) edit a quantitation type, (2) batch define a QT and (3) batch define a QT set. The batch tool, shown in Figure 2.4, allows you to define several ontological data types in a delimited text file and then

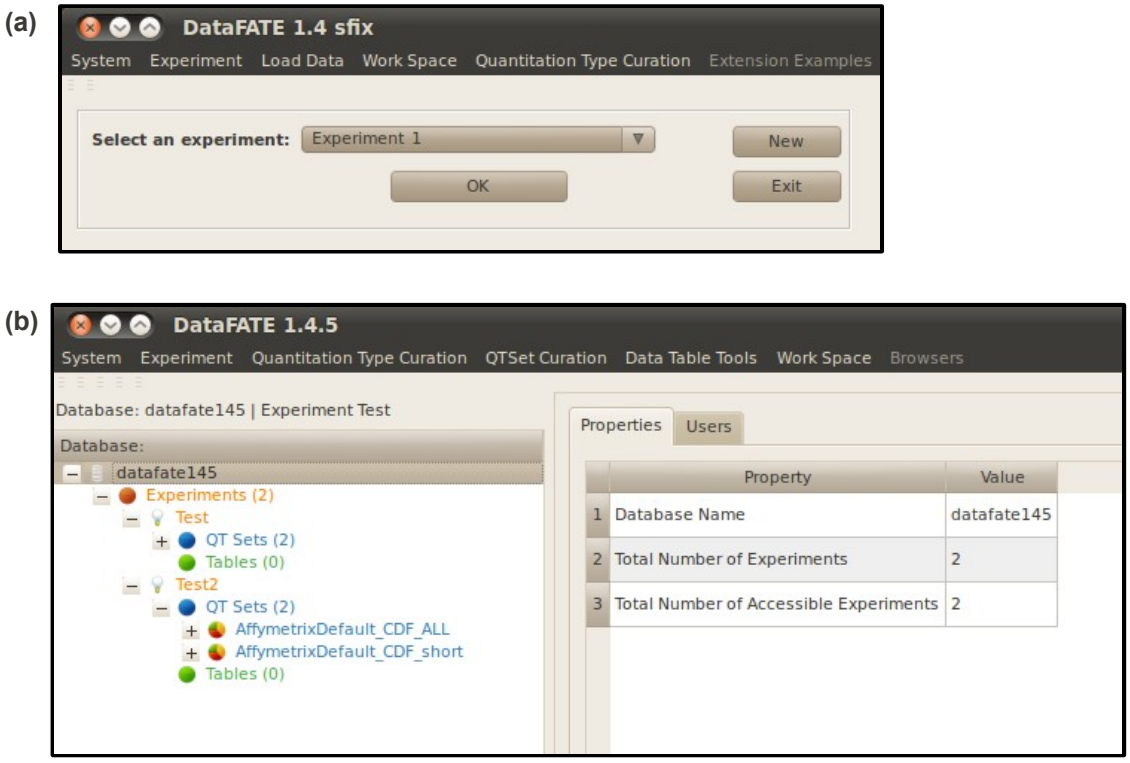


Figure 2.2: Comparison of Data-FATE post-logon screensand navigation. (a) The post-logon screen of version 1.4. (b) The post-logon screen of version 1.4.5. Note the improved method for viewing information about experiments.

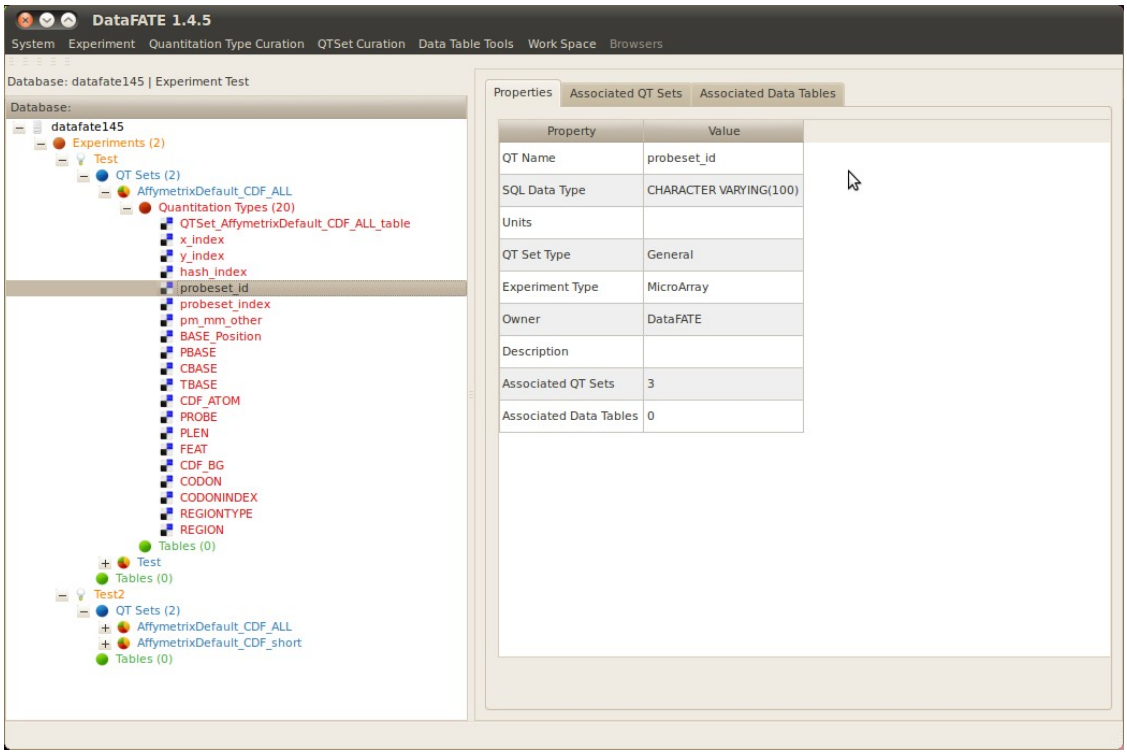


Figure 2.3: Changes to Data-FATE navigation. The enhanced navigation includes hierarchical browsing of databases, experiments and ontologica data types (right) and summary information for the selected type in the dashboard (right), making the SIMS GUI more similar to GUIs developed for relational databases.

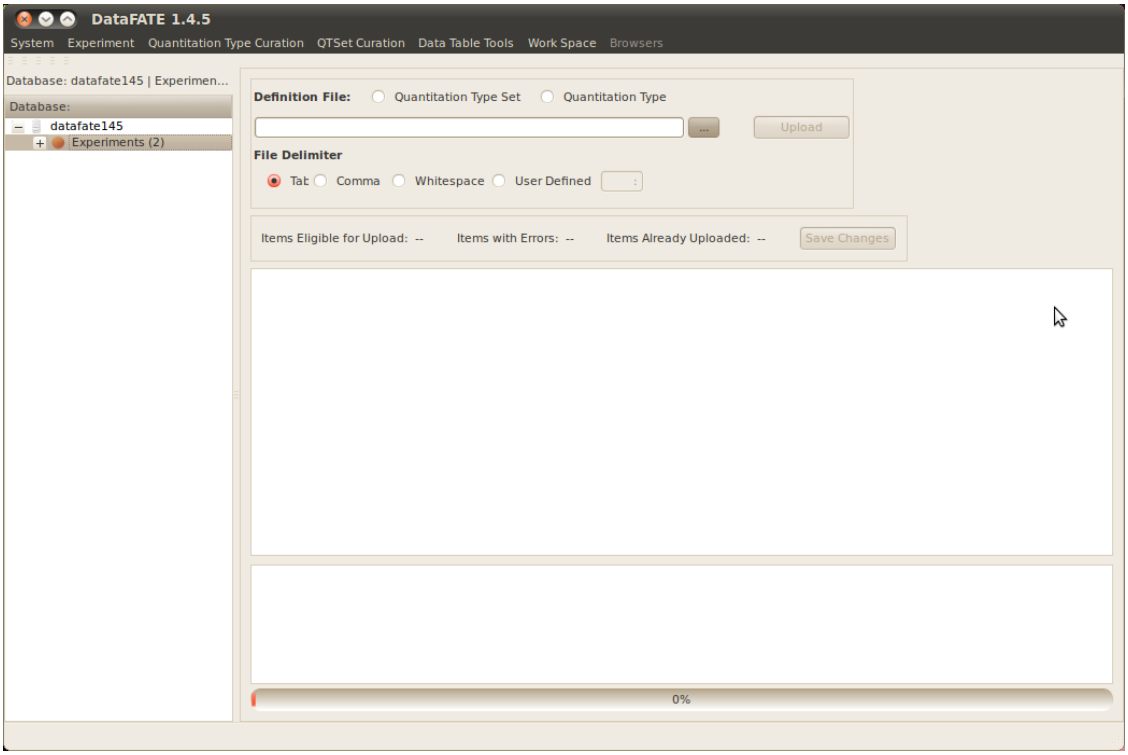


Figure 2.4: Tool for bulk import of ontological data types.

load them all at once.

### 2.4.6 Loading data

Each of the data loading tools from the previous version are included in this one. We added a batch version of the generic loading tool so that you can load multiple data tables at once, shown in Figure 2.5. You specify the data table parameters, e.g. data file, QT set, etc., as needed to create each data table and then run it all at once.

### 2.4.7 Tracking workflows

We designed and implemented a set of tables to allow procedure tracking, to create workflows. The tables allow you to define and save information about user-created protocols (workflows), which consist of a series of data transformations, using SQL, and called analysis scripts (procedures). In this prototype the supported languages for analysis scripts are Python and R. The goal was to show the feasibility of including workflow tools within the SIMS environment.

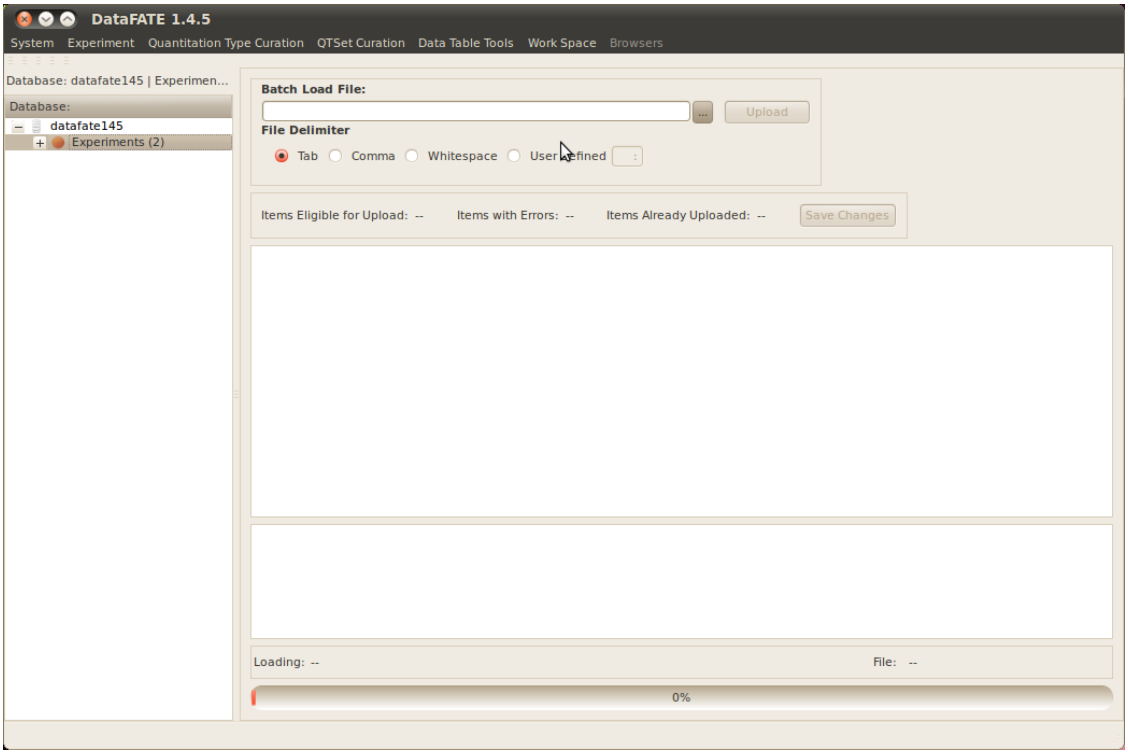


Figure 2.5: Tool for bulk import of data.

### 2.4.8 Testing

While re-designing the system, we tested it using a number of different types of data, such as CEL and CDF files for different versions of Affymetrix GeneChips. We also used it to develop the prototype of the ProbeSieve study (Chapter 4), including probe-genome alignment data for two Affymetrix E. coli arrays and multiple E. coli genomes (including K12) and many raw data files (CEL files). These data sets were the foundation for the research described in the ProbeSieve study, as they informed its eventual design. In the end, we chose not to use Data-FATE for ArrayInitiative (Chapter 3) and ProbeSieve (Chapter 4) because the funding ended, we lost our lead developer, and in the end, we realized that this is probably a 5-person, 5-year project, resources that were not available. Below are the features that would be improve the system should funding resume.

## 2.5 Future work

While the current version of the SIMS is a significant improvement over the previous, it's still a prototype and requires considerable additional work (and funding) to become a mature system. Here we discuss the long-term and short-term goals for its future development.

### Short-term

The short-term goals for the SIMS are geared towards simplifying the user experience, by making the tools more intuitive, or just simplifying them, and hiding the implementation details from the user as much as possible. The point is to allow the researcher to focus on development of useful data models, not with the implementation of the SIMS. In practice, we plan to simplify and make the tools more intutive by using as little relational database jargon as possible.

Towards that end, is became clear that we need to develop simpler methods for defining QTs, for example, by providing simpler relational data type options for researchers who are unfamiliar with those selections. We also want to simplify the process of defining QTSets and for modifying the definition of a QTSet. The latter requires the SIMS to automatically and invisibily change table definitions to reflect changes to a QTSet. We plan to add tools for creating primary keys and indexes and for defining hierarchial and non-hierarchical relationships between QTSets, especially to handle with correct logic those that don't strictly conform to relationships defined by the relational model. Finally, we wish to address the large number of data tables produced when data is imported. Currently, you must know how the SIMS partitioned your data into tables and then run queries directly against them. This becomes difficult to manage when you have hundreds or thousands of tables, which happens quickly with large microarray experiments. To address this problem, we plan to extend SQL to support writing and running "cross-table" queries, which will automatically find and join the appropriate tables based upon your criteria. Concep-

tually, we want users to think that the data for a specific QTSet is stored in the same table, even though it's actually partitioned, and be able to write queries under this assumption. This will greatly simplify using the SIMS.

Long-term

The long-term goals for the SIMS are centered around introducing new tools and functionality. For the descriptive ontologies that researchers define, we plan to develop a tool that automatically extracts ontologies from your experiments, visualizes them, finds similarities and differences between pairs of ontologies and suggests how to merge them without losing important structure and, finally, a tool to share ontolologies. We also plan to introduce data versioning and further develop workflows, with their supporting tools.

# CHAPTER 3: ARRAYINITIATIVE

## 3.1 Introduction

Probes on a microarray represent a frozen view of a genome, quickly outdated when new sequencing studies extend our knowledge and resulting in significant measurement error when analyzing any microarray experiment. There are several bioinformatics approaches to improve probe assignments, but without in-house programming expertise, standardizing these custom array specifications as a usable file (e.g. as Affymetrix CDFs) is difficult, owing mostly to the complexity of the specification file format. However, without correctly standardized les there is a signicant barrier for testing competing analysis approaches since this le is one of the required inputs for many commonly used algorithms. Since our goal in Chapter 4 (ProbeSieve) is to investigate the effects of physical and technical factors on the interpretation of a microarray experiment and to test both gene- and transcription unit-level groupings of probes, we needed to first develop a tool for creating and managing custom array specifications. This led us to develop ArrayInitiative, a tool that simplies the task of creating custom CDFs.

ArrayInitiative is a standalone, cross-platform desktop application for creating correctly formatted, custom versions of manufacturer-provided (default) array specifications, requiring only minimal knowledge of the array specification rules and file formats. Creating a custom array specification requires only minimal knowledge of a manufacturer's specification standards (file formats and logical rules) and the ability to create a simple delimited or XML file. Using ArrayInitiative, users can import default array specifications, import probe sequences for a default array specification,

design and import a custom array specification, export any array specification to multiple output formats, export the probe sequences for any array specification and browse high-level information about the microarray, such as version and number of probes. The initial release of ArrayInitiative supports the Affymetrix 3' IVT expression arrays we currently analyze, but as an open source application, we hope that others will contribute modules for other platforms..

To illustrate the value of re-defined probesets, we then developed a case study where we examine the effects of removing faulty probes from Affymetrix's HG-U95Av2 3' IVT expression array (human). We first identified faulty probes on this array as determined by two published filtering criteria: BaFL pipeline and sequence motifs identified by Upton *et al.* We next defined three different logical designs for the array; the first one removing probes flagged as faulty by BaFL, the second removing probes with Upton sequence motifs, the third removing probes flagged by either BaFL or Upton criteria - the union of the two sets. Using ArrayInitiative, we then created a custom CDF for the three new logical designs. Using these custom CDFs and the default CDF, we summarized the probe set intensities for twenty arrays from a data set generated from human lung adenocarcinoma samples. To summarize the arrays, we used three popular techniques - MAS 5.0, dChip and RMA. For each summarization technique, we compared the probe set intensities summarized using the default CDF against those summarized using the three custom CDFs. We found that the modified specifications significantly changed the summarized expression values as compared to the default, regardless of the summarization technique.

## 3.2 Application overview

ArrayInitiative is a rich client application for creating custom array specifications built upon a default array specification. The default array specification is typically the one provided by the manufacturer and the custom array specification is a user-modified version of that default. Users can: (1) import default array specifications, (2)

import probe sequences for the default array specification, (3) import a custom array specification, (4) export any array specification to multiple output formats (5) export the probe sequences for any array specification and (6) browse high-level information about the array, such as version and number of probes. This release of ArrayInitiative supports Affymetrix 3' IVT expression arrays, and all of the subsequent subsections will assume this type of array.

ArrayInitiative's default main window, shown in . 3.1, consists of an array specification browser, a dashboard and a main menu.

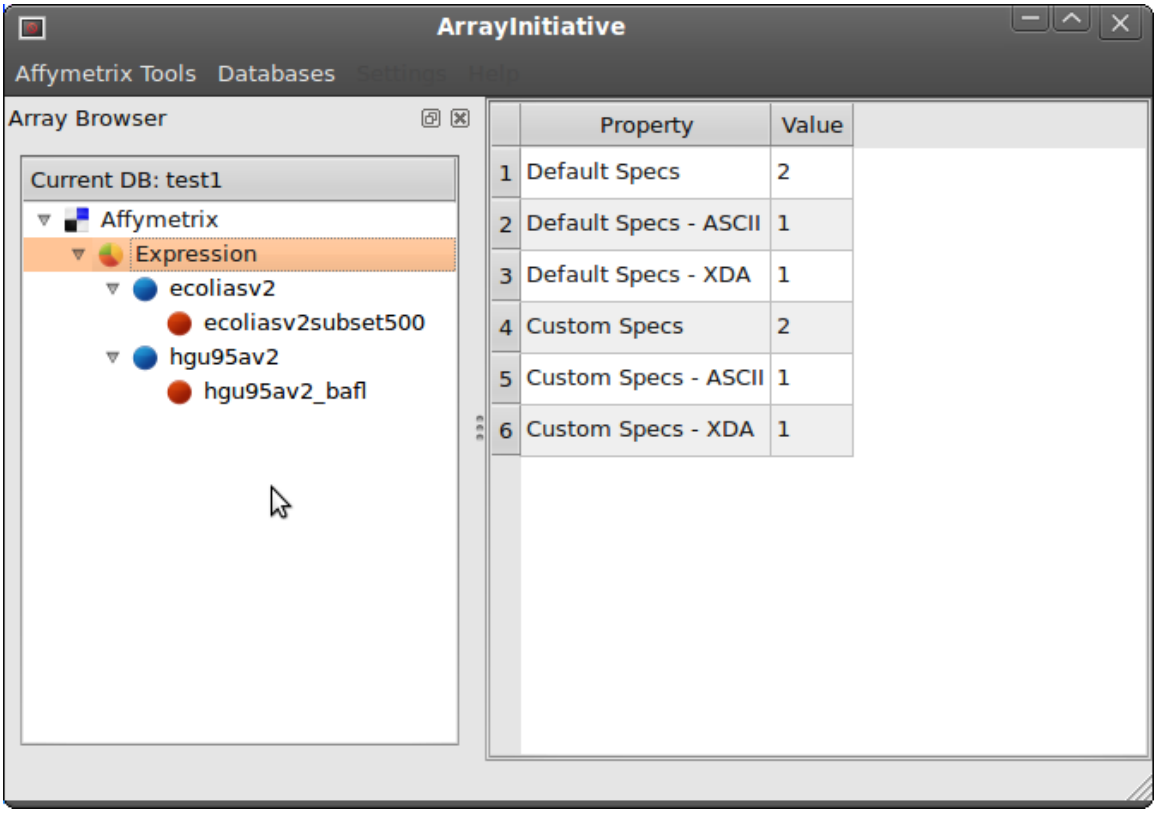


Figure 3.1: *ArrayInitiative main screen.* The ArrayInitiative main screen, consisting of an array specification browser, a dashboard and a main menu.

The array specification browser displays a list of a user's array specifications, organized as a hierarchical tree, while the dashboard displays summary information about the currently selected browser item. For example, when the "Affymetrix → Expression" browser item is selected, ArrayInitiative shows how many default and

custom Affymetrix 3' expression array specifications there are; when a user clicks on an array specification in the browser, summary information for that specification is displayed in the dashboard. All of ArrayInitiative's tools, such as the one for importing a default array specification, can be opened from either the main menu or from context-sensitive (right-click) menus available in the specification browser. Each of the tools in ArrayInitiative open as modal dialog windows.

### 3.2.1 Implementation

We developed ArrayInitiative as a standalone, rich client desktop application with an integrated backend database. The user interface was implemented with PyQt [107], a Python [105] binding of QT from Riverbank Computing. We used SQLite [108] as the backend database, as implemented in Python's *sqlite3* module [109], because it requires minimal installation/setup, administration and maintenance tasks for the user and is a standard library module in Python 2.5+. Each of the main components are cross-platform and freely available. ArrayInitiative can be downloaded from the "Downloads" section at `http://wellerlab.uncc.edu/ArrayInitiative/index.html`.

### 3.2.2 Functionality

#### 3.2.2.1 Context-sensitive (right-click) menus

The array specification browser gives right-click access to the main menu items; the resulting form values are pre-populated based on the current browser selection. Renaming and deleting array specifications can only be done using the context menu.

#### 3.2.2.2 Creating and managing multiple databases

When first using ArrayInitiative, users will need to create at least one database before they can access any of array-specific functionality of ArrayInitiative. Multiple ArrayInitiative databases can be created to logically separate distinct sets of arrays, if desired. In addition, users can update the information for an existing database and switch between databases by setting the active database.

#### 3.2.2.3 Importing a default array specification

Users can import the array specification (probe set definitions) for an array from a CDF file (ASCII, versions 3-5 and XDA, versions 1-3). Usually, users will import a default array specification from a CDF provided by Affymetrix, but they can also import a default array specification from a custom CDF instead. Users must import at least one default array specification before importing custom array specifications and writing custom CDFs.

#### 3.2.2.4 Importing probe sequences

After importing a default array specification, users can import the probe sequences for a default array specification, using the FASTA or tab-delimited probe sequence file provided by Affymetrix for that array. ArrayInitiative will automatically generate the missing mismatch probe sequences. See the "File Formats" section of the manual — available online in the supplementary site — for details about the supported formats for a probe sequences file.

#### 3.2.2.5 Creating a custom array specification file

After importing a default array specification, users can create a custom array specification for any imported default array specification. When creating a custom array specification file to import, users can instruct ArrayInitiative to copy an existing probe set, re-define an existing probe set or define an entirely new probe set. When defining, or re-defining, a probe set, users can use any of the probe pairs from the default array specification. ArrayInitiative treats probe pairs as atomic units, and as such, users can't add just the PM or MM probes to a probe set definition. Currently, ArrayInitiative accepts a full specification file type (delimited or XML), requiring that users explicitly define every probe set. See the "File Formats" section of the manual — available online in the supplementary site — for details about the supported formats of a custom array specification.

#### 3.2.2.6 Importing a custom array specification

After creating a full specification file, users can import them into ArrayInitiative. Users can define multiple custom versions for any default array specification.

#### 3.2.2.7 Exporting an array specification

Users can export default and custom array specifications as a CDF (ASCII or XDA), an XML file or a delimited file. See the "File Formats" section of the manual — available online in the supplementary site — for details about the output types.

#### 3.2.2.8 Exporting probe sequences for an array specification

Users can export the probe sequences for a default or custom array specification as a FASTA, XML or delimited file. See the File Formats page for details about the output types. When exporting a custom array specification as a CDF, the type — ASCII or XDA — will be the same as the parent default array specification.

## 3.3 Case study

In this section, we illustrate why ArrayInitiative is useful to microarray researchers by presenting a case study in which we create custom CDFs based upon two different, published probe-filtering techniques and then use Bioconductor algorithms to investigate the effect of the probe set re-definitions on the summarized expression values. The complete case study code, data and results are available under the "Downloads" section at `http://wellerlab.uncc.edu/ArrayInitiative/index.html`.

### Introduction

Imagine that you, as a researcher who is reasonably proficient with programming, discover two different probe-filtering techniques for Affymetrix arrays while reading the literature. Both of them seem reasonable and you think that, by incorporating such QC steps, you could improve the results you get when analyzing your expression arrays with Bioconductor tools. Since your favorite Bioconductor packages require a well-formed CDF, you search the web to see if someone has created a custom CDF based upon both filters. Unfortunately, you can't find one and must generate the

custom CDF yourself or rewrite and test several complicated algorithms. Convinced that the filters will improve your results, you decide to create a custom CDF from scratch. The developers of technique A conveniently provide a comma-delimited text file with the new probe set definitions for the array type that you're interested in, while the developers of technique B provide a custom CDF with their filter, also for your array type. You then need to compare the two different probe set definitions to make sure they don't conflict and then merge their individual probe set definitions into a single custom CDF. Examining the delimited files is relatively straightforward, so filter A's probe set definitions are already usable; however, to get the probe set definitions for filter B, you need to parse the rather complex CDF file. After some time and effort, you manage to learn the CDF format and successfully retrieve the probe set definitions for filter B. With some coding magic, you create a joint probe set definition that is the intersection of the two filters. Confident in your knowledge of the CDF format, you write some code to create the custom CDF, which eventually is accepted by the analysis packages after much trial-and-error. Upon analyzing your arrays, it appears that, indeed, the two filtering techniques, in tandem, significantly improve your results. Excited by your success, you want to apply the same probe-level filters to an expanded set of arrays, some of which were done on a later version of the array. As you acquire the necessary files you realize that the later version of the array is described by a different kind of CDF, in the XDA format, which is entirely different from the CDF format that you learned. Dispiritedly, you set out to learn another format and start the process over again.

Not only is the above scenario likely, it is also fairly optimistic. Many research labs do not have the in-house computational expertise to create custom CDFs easily, nor should every lab be required to learn about the CDF formats to reap the benefits of research into probe-level filters on Affymetrix microarrays. This is exactly why a generic custom CDF creator like ArrayInitiative is useful.

The case study presented here illustrates the merging of two real sets of probe filters, that we term 'BaFL' and 'Upton' (described more fully below). We created custom HG-U95Av2 CDFs for each of them and then used three different Bioconductor packages — RMA, dChip and MAS 5.0 — to determine the independent and joint effect of each filter. Lest the reader be unconvinced that such filters would alter the outcome, for a given custom CDF and summarization method, we compared the probe set intensities calculated using the custom CDF to those calculated using the default CDF.

### 3.3.1 The HG-U95Av2 microarray and the Bhattacharjee data set

The 'Bhattacharjee' data set, which contains data for arrays reporting on 139 distinct macro-dissected human lung adenocarcinoma samples, was assayed using 190 HG-U95Av2 arrays[110]. Of these, 47 samples had 2-4 replicate arrays (most have only two). The HG-U95Av2 array has 12,625 probe sets and 201,800 probe pairs (403,600 probes), with most probe sets having 16 probe pairs (32 probes). The full distribution of probe pairs per probe set is presented in Supplementary Table 1.

For this case study, we analyzed twenty randomly selected arrays (RAND) from 190 Bhattacharjee adenocarcinoma arrays, shown in Supplementary Table 2. When selecting the arrays, we excluded any arrays that exhibited array-wide technical problems, as identified by Thompson *et al.*[99], from the sample pool.

### 3.3.2 Probe-filtering techniques

#### 3.3.2.1 BaFL

Thompson *et al.* [99] developed a "white box" pipeline – Biologically applied Filter Levels (BaFL) – to identify and filter microarray probes that are likely to report incorrect or misleading intensities based upon certain biological properties, such as the presence of SNPs in the probe sequence (e.g. as identified by AffyMapsDetector [100]), probe cross-hybridization, internal structure in either the probe or target sequence that reduces binding affinity, and probe intensities that fall outside the linear range

of the scanning device **[81, 82]**. Thompson *et al.* provided comma-delimited files of filtered (deprecated probes removed) probe set definitions for the HG-U95Av2 and HG-U133 array types.

#### 3.3.2.2 Upton

Upton *et al.* [79, 80] reported that probes with certain sequence motifs have intensities that are uncorrelated with the other probes in the same probe set; however, they tend to correlate well with any probes having the same sequence motif, regardless of probe set membership. In this case study, we will focus on the two major types of problematic sequence motifs identified by Upton *et al.* [80]: G-runs and primer spacers. Probes with the G-run motif, $\geq 4$ Gs in a row, tend to produce consistently high intensities, with some position dependence. The primer spacer motif, CCTCC, is related to the incorporation of a T7-binding site when a mRNA is amplified during target preparation. When the target is amplified in this manner, the probe intensities tend to be higher than most, introducing a spurious correlation similar to that seen with G-runs. Since both of these sequence motifs introduce a systematic bias when summarizing probe set intensities, any probes including them should be removed from a CDF prior to calculating expression values. This is always true for the G-run motif and is true for the primer spacer motif when the target is amplified by incorporating a T7-binding site. The reports by Upton et al. provided good insights about identifying problematic probes, but they did not provide a modified CDF, a flat-file of probe set definitions nor a list of deprecated probes for a given array version.

### 3.3.3 Are the probe-filtering techniques independent?

When different groups develop QC filters independently there may be overlap or conflicts of which they are unaware. Therefore, before proceeding with creating custom CDFs and downstream analysis, we first assessed the overlap between the BaFL and Upton filter sets to determine if they are truly independent filters.

Fig. 3.2a shows how many *probe pairs* were removed independently and jointly

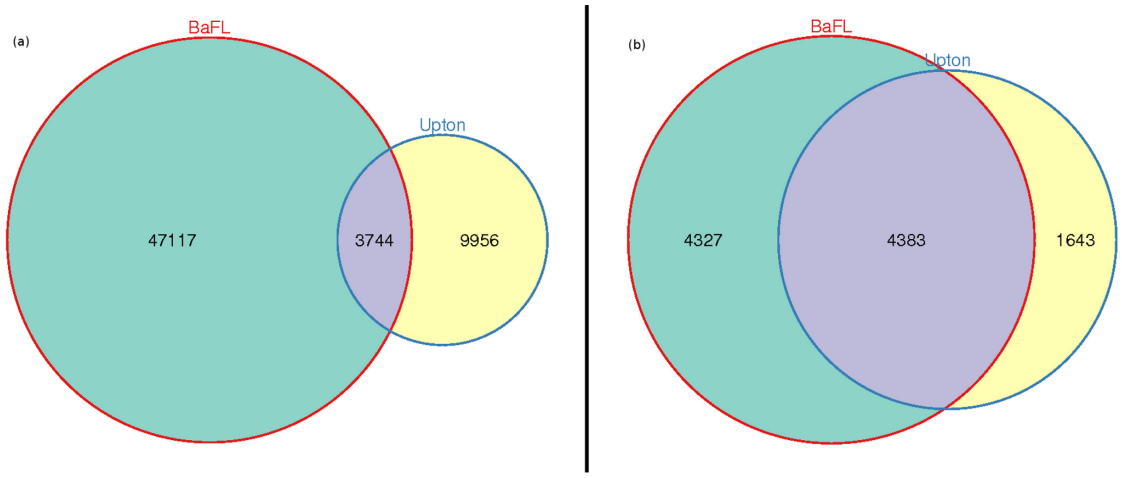


Figure 3.2: *Independent and joint effects of the BaFL and Upton filter sets. (a)*: The total number of probe pairs removed by either the BaFL or Upton filter sets was 56,994/201,800 (28.2%). The Venn diagram shows the number of probe pairs removed only by the BaFL filter set (blue), the number of probe pairs removed by the Upton filter set (yellow), and the number of probe pairs removed by both filter sets. *(b)*: The total number of probe sets removed or modified by either the BaFL or Upton filter sets was 9,799/12,625 (77.6%). The Venn diagram shows the number of probe sets affected only by the BaFL filter set (blue), the number of probe sets affected only by the Upton filter set (yellow), and the number of probe sets affected by both filter sets.

by each of the filter sets and Fig. 3.2b shows how many *probe sets* were modified or removed independently and jointly by each of the filter sets. Most analyses are performed on probe set data, but the affected probe pairs are not necessarily homogeneously distributed, so we examined both aspects. Based upon these results, we see that the two filters generally operate on different sets of probe pairs, with minor overlap (4.1%). The BaFL filter set removes a significantly larger number of probe pairs than the Upton filter set. We also see that the two filter sets jointly affect a large fraction of the probe sets (31.1%), although a significant portion of them are independently affected by each filter. The greater overlap between the two filters in the latter case is expected since each probe set consists of multiple probe pairs.

### 3.3.4 Creating the custom CDFs

We created three custom CDFs using ArrayInitiative: a BaFL-only custom CDF, an Upton-only custom CDF and a BaFL plus Upton joint CDF. Each filter set required a unique approach for generating the probe set definitions due to the different ways that they were communicated; however, after we defined the probe sets for each filter set, the steps for creating the custom CDFs were identical. Fig. 3.3 shows a graphical summary of the CDF creation workflow.

The first actions were common steps. We created a new ArrayInitiative database and imported the default HG-U95Av2 CDF from the file provided by Affymetrix. Next, we imported the PM probe sequences using the tab-delimited file provided by Affymetrix and instructed ArrayInitiative to automatically generate the corresponding MM probe sequences. Finally, we exported the default CDF probe set definitions (with probe sequences) as a comma-delimited text file. When generating the custom probe set definitions in the subsequent steps, we queried the ArrayInitiative database directly for information about the default probe set definitions as querying databases tends to be more efficient and straightforward than searching for information in flat files. The end-point of the custom probe set definition stage was to have in hand comma-delimited file (CSV) with the following columns per line: (1) probe set ID, (2) PM probe ID, (3) x-coordinate of the PM probe, (4) y-coordinate of the PM probe, (5) MM probe ID, (6) x-coordinate of the MM probe and (7) y-coordinate of the MM probe. In order to keep the method comparison fair we required that each probe set have at least 4 probe pairs remaining; if it did not, we removed it before creating the custom CDFs.

When creating the BaFL probe set definitions, we started with the comma-delimited filtered probe set definitions for the HG-U95Av2 array provided by Thompson *et al.* Since that probe set definition file included only the PM probes, we first queried the ArrayInitiative database to get the full probe set definitions of the de-

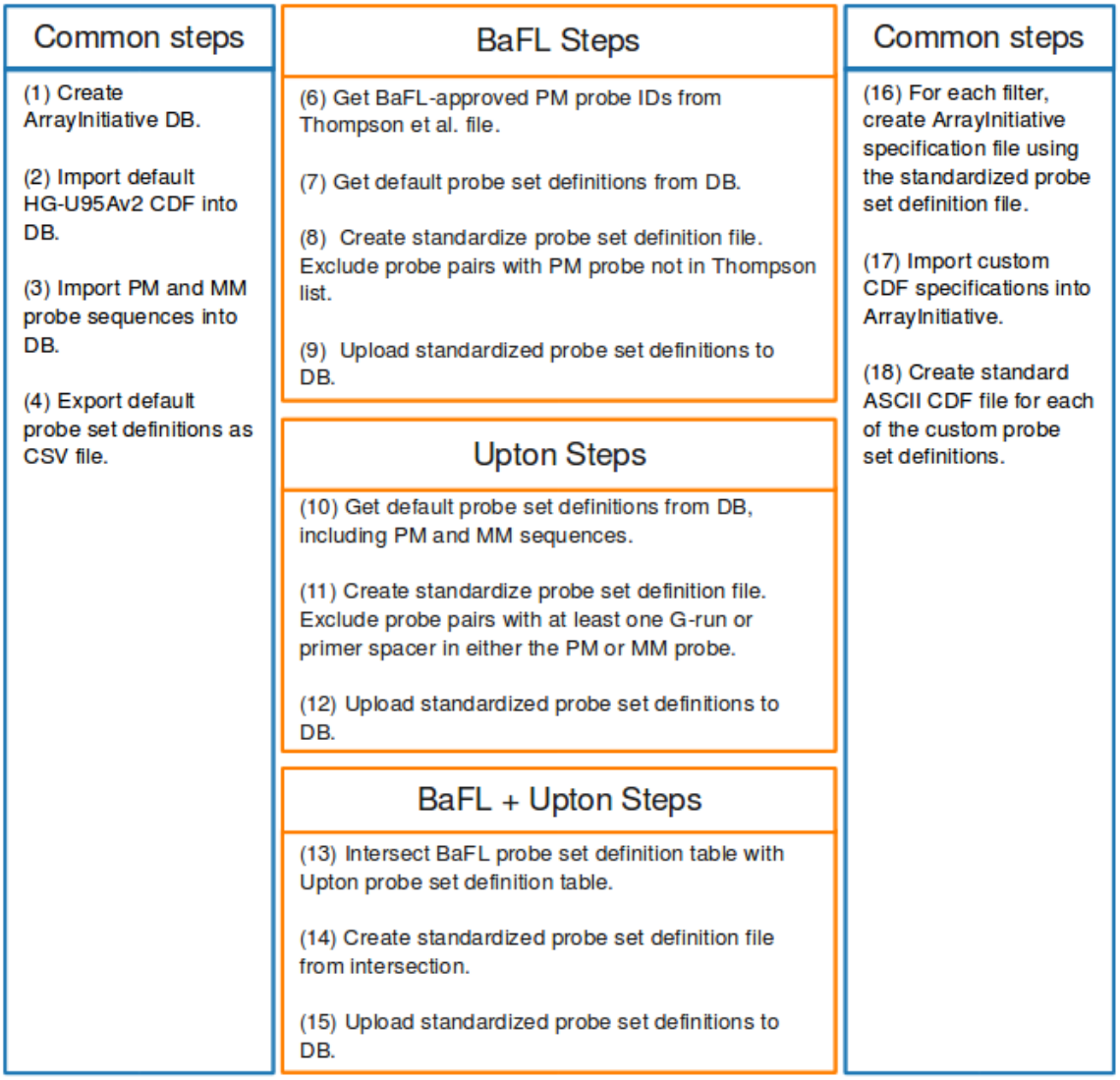


Figure 3.3: *Workflow for creating the custom CDFs.* Workflow for creating the custom BaFL, Upton and BaFL + Upton custom CDFs. The boxes in blue are common steps while the boxes in orange are steps unique to a particular filter set.

fault CDF. When creating the standardized probe set definition file, we included only those probe pairs whose PM probe was in the author's probe set definitions. We then uploaded the standardized probe set definitions into the ArrayInitiative database.

Creating the Upton probe set definitions was somewhat trickier because we needed first to identify the G-run probes on the HG-U95Av2 array. Again, we first queried the ArrayInitiative database to get the full probe set definitions of the default CDF, including the PM and MM probe sequences. When creating the standardized probe set definition file for this filter, we identified probe pairs — using regular expressions — that had at least one G-run or primer spacer in either the PM or MM probe sequence and then excluded that probe pair from the final probe set definition. We then uploaded the standardized probe set definitions into the ArrayInitiative database.

Since the BaFL + Upton CDF is the intersection of the probe pairs that survived the BaFL and Upton filters, we retrieved the joint probe set definitions from the ArrayInitiative database by intersecting (standard 'INTERSECT' SQL statement) the BaFL probe set definition table and the G-run probe set definition table (created in the previous steps). Now having a list of the surviving probe pairs, we created a standardized probe set definition file and uploaded this data into the ArrayInitiative database.

Having standardized probe set definition files for each of the probe filters, the final steps for creating a custom CDF for each are identical. We first created ArrayInitiative specification files for each of the filters using the standardized probe set definition files and then imported the custom CDF specifications into ArrayInitiative. Finally, we created a standard ASCII CDF file for each of the custom probe set definitions in ArrayInitiative.

Table 3.1 shows how the custom CDFs were changed relative to the original and Fig. 3.4 compares the frequency with which the indicated number of probe pairs are removed from probe sets for each of the three custom CDFs (e.g. the number of

Table 3.1: *Filter set modifications to the HG-95Av2 specification.* Modifications to the default HG-U95Av2 specification made by each set of filters.

| CDF name | Removed probe sets | Modified probe sets | Unmodified probe sets |
|---|---|---|---|
| Upton filter set | 1 | 4,083 | 8,303 |
| BaFL filter set | 1,406 | 7,125 | 3,856 |
| BaFL + Upton filter set | 1,460 | 8,570 | 2,357 |

probe sets with zero probe pairs removed, one probe pair removed, two probe pairs removed, etc.)

### 3.3.5 Creating and validating Bioconductor CDF packages

Many of the Bioconductor packages aimed at analyzing Affymetrix arrays use a specialized R package representation of a CDF instead of the actual CDF; there are pre-generated packages for many of the default CDFs. Since we are using custom CDFs for downstream analysis, we first created and installed our own R packages for the three custom CDFs generated by ArrayInitiative, as follows:

1. Made the packages using the *make.cdf.package* function in the *makecdfenv* package [111].

2. Installed the custom CDF packages using *R CMD INSTALL*.

With the custom CDF packages successfully installed, we compared, for each filter, the probe set definitions in the existing R packages with the probe set definitions in ArrayInitiative, as follows:

1. Exported Bioconductor's internal probe set definitions for the custom CDFs – using the *ls* and *get* R functions – to a set of delimited files and then uploaded the data to the ArrayInitiative database (three tables total).

2. Verified that the number of Bioconductor probe pairs equaled the number of ArrayInitiative probe pairs (SQL 'COUNT').

3. Verified that the member probe pairs in Bioconductor were the same as the member probe pairs in ArrayInitiative (SQL 'INTERSECT').

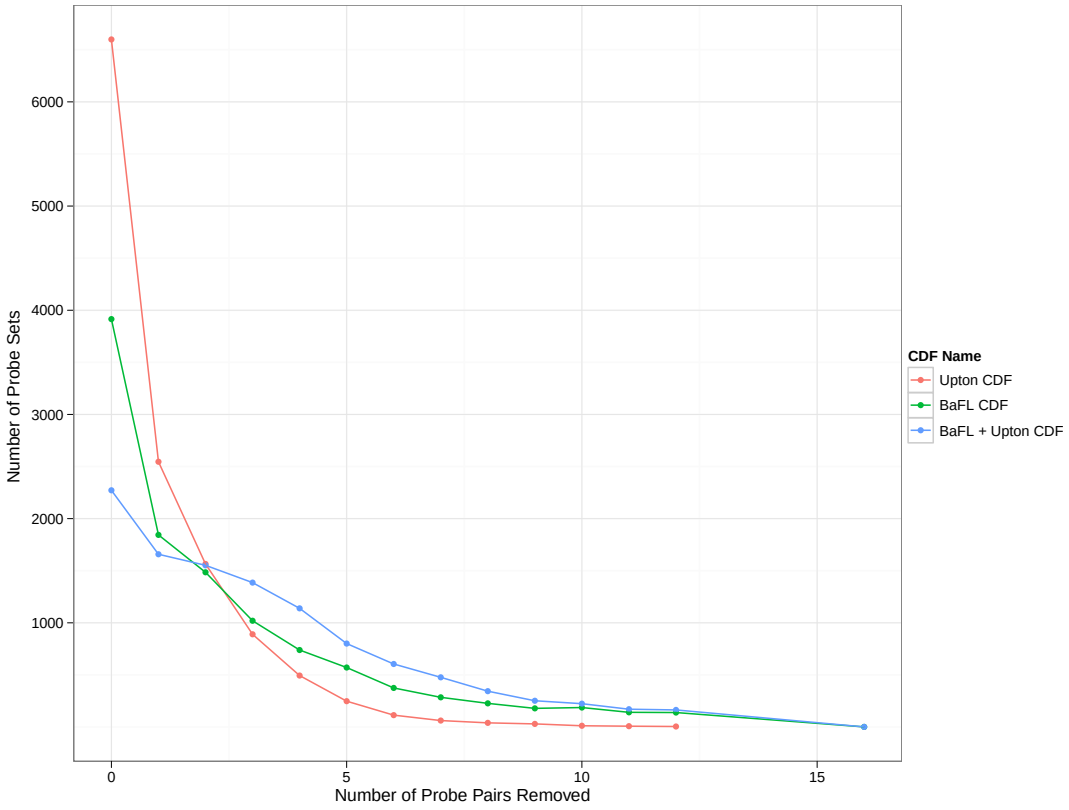


Figure 3.4: *Number of probe pairs removed by individual filter sets.* Summary of number of probe pairs removed from standard probe sets — those having 16 probe pairs — by each of the three filter sets. Presented for each custom CDF are the total number of probe sets that survived the cleansing process. Default CDF = 12,387 standard probe sets, Upton CDF = 12,386 standard probe sets, BaFL CDF = 10,981 standard probe sets, BaFL + Upton = 10,927 standard probe sets.

Using the above procedure, we verified that the probe set definitions for each filter were consistent between Bioconductor and ArrayInitiative, showing that ArrayInitiative-generated CDFs are compatible with one of the most widely used microarray analysis packages. Since the probe set definitions were consistent, we can reasonably assume that any differences in downstream analysis will be the result of the custom probe set definitions, not from misinterpreting set membership.

### 3.3.6 Differences in summarized probe set intensities

How do the BaFL and Upton filter sets independently, and jointly, affect summarized probe set expression values? For the three summarization methods chosen (MAS 5.0, dChip, and RMA), we determined how, on average, the custom expression values changed with respect to the default expression values as we removed probe pairs.

We only analyzed the 12,387 probe sets with 16 probe pairs in the default CDF (henceforth called standard probe sets) and only removed from 0 to 12 probe pairs, so that at least 4 remained to a set. We did this for several reasons: (1) standard probe sets represent the vast majority of those on the array and most are designed to interrogate transcripts, (2) the majority of non-standard probe sets represent the minority of those on the array and most are designed for diagnostic or quality control purposes and (3) we wanted to use a consistent probe set size to eliminate that as a factor when analyzing the downstream effect on expression values.

For each unique combination of summarization method and RAND array, we calculated the expression values of the standard probe sets using the default and custom probe set definitions. Then for each probe set, we calculated the percent change between the expression values, as follows:

$$\Delta = \frac{E_c - E_d}{E_d} * 100$$

where $E_c$ is the custom expression value and $E_d$ is the default expression value. For

each distinct combination of summarization method and custom CDF, we calculated the average delta, across all of the RAND arrays, as we removed probe pairs. The workflow is depicted in Supplementary Fig. 1.

Before running the analysis, we postulated that the Upton filter set would decrease probe set intensities as we removed probe sets, since probes with G-runs and primer spacers tend to have a much higher intensity than other probes in the probe set; we expected that the BaFL filter set would increase the probe set expression values as we removed probe sets because its filters tend to remove low intensity PM probes; we expected that the joint filter probe set expression values would fall between those produced by the two independent filter sets, but heavily weighted towards the BaFL probe set expression values, since it removed many more probe pairs.

3.3.6.1 MAS 5.0

Fig. 3.5a shows the average expression changes seen when we used MAS 5.0 to summarize the probe sets. The Upton filter set influenced the probe set expression values in two distinct ways: when we removed 1-5 probe pairs, the expression values stayed relatively constant compared to the default CDF; when we removed 6-9 probe pairs, the expression values increased (except at 7); when we removed 10-12 probe pairs, the expression values decreased. This result was surprising since we expected the probe set expression values to consistently decrease. The BaFL filter set consistently resulted in increased probe set expression values as we removed probe pairs, while the joint filter set was a blend of the two independent filter sets, although heavily weighted towards the BaFL filter set.

3.3.6.2 dChip

Fig. 3.5b shows the average expression changes seen when we used dChip to summarize the probe sets. The Upton filter set decreased the probe set expression values, but exhibited somewhat erratic behavior. The BaFL filter set, in general, decreased the probe set expression values, reaching a maximum positive change at

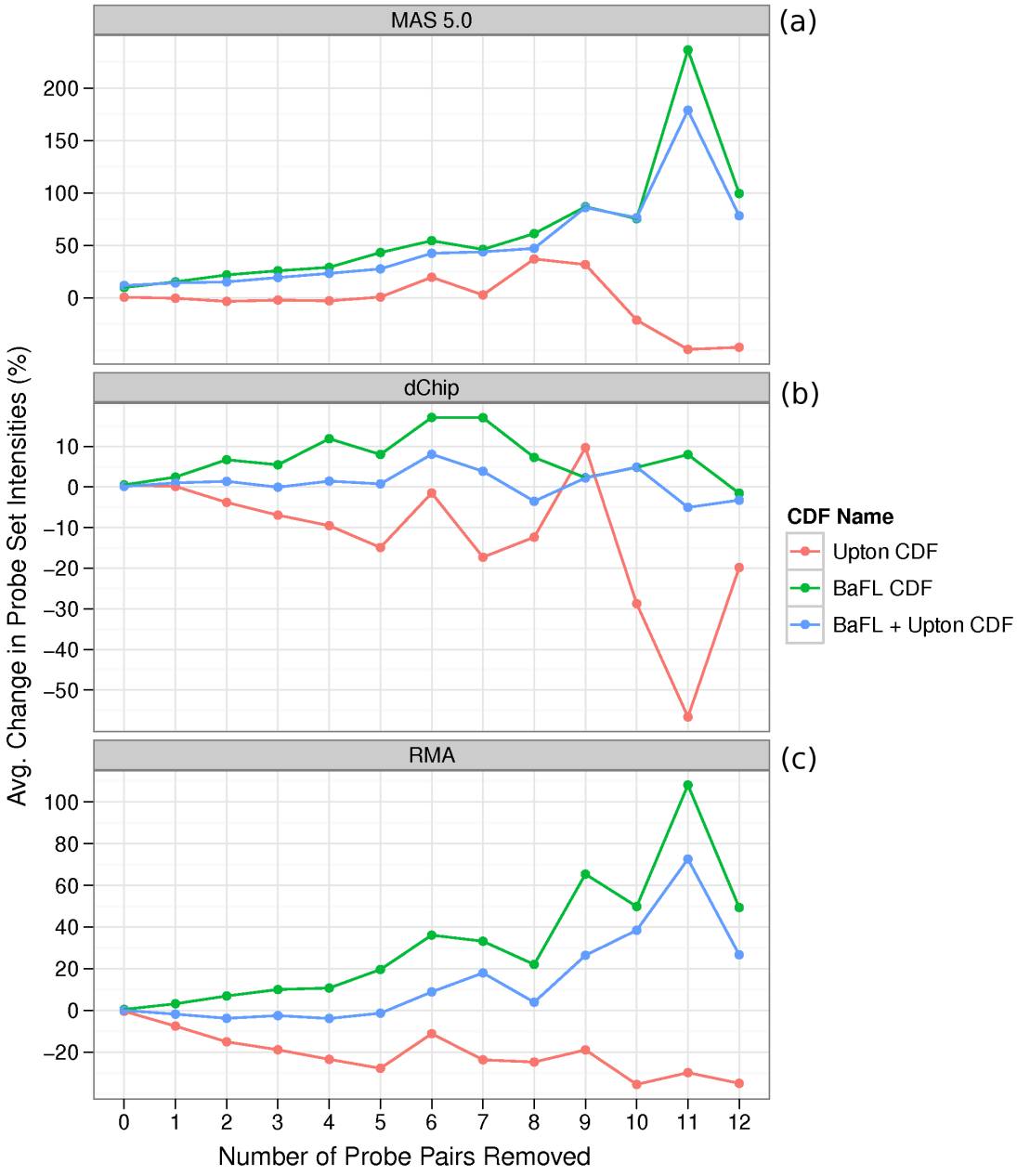


Figure 3.5: *Difference between summarized probe set intensities.* Probe set intensities were summarized by MAS 5.0, dChip and RMA for each of the three custom CDFs and for the default CDF. The graphed lines show the average percent change in custom CDF probe set expression values with respect to the default CDF expression values as we removed probe pairs. *(a)* Probe set intensities were summarized by MAS 5.0. *(b)* Probe set intensities were summarized by dChip. *(c)* Probe set intensities were summarized by RMA.

6-7 probe pairs removed. The expression values decreased when we removed 12 probe pairs. The joint filter set was a blend of the two independent filter sets, only somewhat weighted towards the BaFL filter set.

#### 3.3.6.3 RMA

Fig. 3.5c shows the average expression changes seen when we used RMA to summarize the probe sets. The Upton filter set consistently decreased the probe set expression values while the BaFL filter set consistently increased the expression values. The joint filter set was a blend of the two independent filter sets: the values were slightly weighted towards the BaFL filter set when we removed 1-6 probe pairs and heavily weighted towards the BaFL filter set when we removed 7-12 probe pairs.

### 3.3.7 Case study discussion

The Upton filter set decreased the probe set expression values when they were summarized by dChip and RMA, a trend not observed when we summarized the probe sets with MAS 5.0. The MAS 5.0 expression values were unresponsive to the Upton filters when we removed 1-5 probe pairs, while its effect was fairly erratic in the 6-12 range. The BaFL filter set consistently increased the probe set expression values for all of the summarization methods, with MAS 5.0 and RMA being particularly responsive. The joint filter set produced intermediate expression values that were a blend of the two independent filter sets when summarized with either dChip and RMA; the effect was generally additive. The BaFL filters had a stronger influence on the expression values, but this is not surprising, given that the BaFL filter set removed significantly more probe pairs than the Upton filter set. When summarizing with MAS 5.0, changes in the expression values were largely driven by the BaFL filters, with the Upton filters having little effect.

In considering the joint filter set, RMA exhibited trends in expression value changes that best fit our prior expectations. Considering the *magnitude* of expression value changes, the joint filter changed the MAS 5.0 expression values the most,

followed by RMA and then dChip. While the expression values for MAS 5.0 and RMA changed by factors of 20-100% for many of the data points, the changes seen with dChip were much lower, in the 2-15% range, suggesting that dChip is the least responsive to changes in the probe set definitions.

From these results, we may conclude that the filter sets significantly alter the *value* of the estimated target concentration when using any of the summarization methods, although we can't speculate if it drives the values towards or away from the true value. Also, we note that ArrayInitiative has finally allowed our lab to apply MAS 5.0, dChip and RMA to a BaFL-filtered data set, which has been one our research goals for a while.

## 3.4 Future work

Long-term, we intend to develop an open API that will support module development by external programmers for a large number of array types and manufacturers. For example, the research community might create modules that implement a specific strategy for re-defining probe sets (e.g. gene-specific, transcript-specific, exon-specific, tissue-specific, 3'-end specific) or modules that pre-process and remove probes that contain undesirable sequence motifs, such as runs of Gs. Short-term, our research goals dictate adding support for Affymetrix SNP and exon arrays, adding support for Agilent human 4 x 44k arrays, development of a tool to report just the differences between two CDFs, development of a tool to convert between the Affymetrix ASCII and XDA formats and development of a tool to merge two or more different probe set definitions (union, intersection, difference) for the same array type. We also need a variant of the merging tool that can define consensus probe sets among different, but related, platforms. In particular, we have pooled data from adenocarcinoma studies assayed on four versions of the Affymetrix human genome arrays: HG-U95, HG-U133, HG-U133A and HG-U133 Plus 2.0. These arrays share many same-sequence probe pairs, but the names of their parent probe sets and their location on the arrays are

different. A consensus merging tool will identify the common probe pairs by their sequence and then group them into biologically relevant probe sets. The probe set identifiers and probe sequences will then be consistent across arrays, differing only in probe coordinates. This would require a custom CDF for each array version, but all of them would consistently measure the same subsequences in each transcript. Finally, we intend to add support for a difference specification type, which will allow users to specify a custom CDF as an exact copy of the baseline CDF, except for any explicitly stated differences, most likely useful for those studying only a few genes in great detail.

# CHAPTER 4: PROBESIEVE

## 4.1 Introduction

Microarrays have long suffered from accuracy and reproducibility problems, which has made it difficult to confidently and consistently interpret the results of microarray experiments [74]. Without reasonably accurate and consistent estimates of transcript abundance, it's difficult to model all of the nodes and edges in a transcriptional regulatory network. Especially important for prokaryotes, the estimates of transcript abundance by genes in the same transcription unit should have similar exression profiles across conditions (although the actual values may vary due to binding affinitiy differences) - if not, there is either a problem with the probe sets measuring those genes, or it might indicate the presence of a new transcription unit or that the annotation for the existing one needs to be modified. In any case, to accurately model a TRN, you must be able to trust the measurments that you're using to characterize it.

Although many causes have been identified, an important source is failures in the probe design process, which lead to confounded measurements. A number of known physical and technical factors affect probe behavior, such as probe-to-target mapping problems and secondary structure problems. Researchers have developed both statistics- and factor-based methods to identify and discard probes (or their measurements) that are likely not reporting faithfully. While statistical methods, such as RMA, GC-RMA, dChip and MAS 5.0[83, 84, 85, 86], are easy to use and good at removing generalized measurement error, it's likely that they remove meaningful experimental variation along with the unwanted variation [112]. If you know the

sources of the unwanted variation, it's advisable to remove probes affected by them before trying to interpret the results of microarray experiments. This is the philosophy behind factor-based probe filtering methods. Over the years several have been shown to improve our interpretation of microarray experiments [89].

While an improvement, factor-based approaches have several shortcomings. With the exception of BaFL [99], a consolidated factor-based pipeline doesn't exist. No common data set or test methods exist to evaluate the impact of a factor-based filter, making it difficult to consistently assess the impact of a filter. For example, one study may declare success if an increase in the concordance of differential gene expression lists occurs [ref], while another may use improved sample classification as its measure of success [ref]. To avoid experimental and factor bias, several evaluation methods should be used on several data sets. Other considerations include whether the factors are independent, whether some are irrelevant (having a negligible effect) and whether this is true across platforms and experiments. For example, a SNP filter is not relevant for a prokaryotic organism. Also, while some studies account for binding affinity differences between probes, the handling of those differences is inconsistent – some approaches remove poor-affinity probes altogether while others use a linear model to average signal across a set of probes [85, 83, 113]. A number of studies have examined the relationship between intensity and binding affinity and found it to be complex – however these studies have not considered all of the other factors affecting the intensity. A question that still concerns us is whether some factors remain to be identified. Thompson noted latent structure in the data set used to validate the BaFL pipeline, which was ascribed to unknown factors [99], but could also have reflected sample characteristics. No study presents a method to answer this question. To address several of these limitations, we developed the ProbeSieve pipeline.

The ProbeSieve pipeline builds upon the BaFL pipeline, with several key extensions, modifications and diagnostic methods to tune it for use with experiments done

on Affymetrix E coli antisense expression arrays. The core workflow is similar: enforcing upper and lower limits for raw probe intensities that better reflect the sensitivity range of the scanner and removing probes that cross-hybridize or have no target in the updated genome annotation (both by exact sequence alignments and structural simulations). However, there are several key differences. We removed the filter for SNPs (since they're not relevant to prokaryotes), modified the approach for enforcing scanner sensitivity to allow experiment-specific baseline detection, and included the ability to re-assign probes with misassigned targets (rather than simply deprecate them). When identifying probes that cross-hybridize, or have an unidentified or misassigned target, we consider not only exact alignments, but inexact ones that follow the Kane criteria[78], expanding the range of binding partners considered in the original pipeline. We calculate the binding affinity between all of these probe-target combinations (exact and inexact), giving us a more complete picture of the competitive pool of targets for a probe. We then remove or re-assign probes using this binding affinity, similar to the approach for exact alignments in the BaFL pipeline. Besides using the binding affinity to make a binary decision to keep or remove a probe, we also use it to modulate the reported probe intensities. For example, knowing that a probe binds to a singe target with only a 60% efficiency, we can calibrate the probe's intensity to account for this bias.

To evaluate the impact of the different factors, we created a suite methods (both novel and published) including: affiliated probe correlation, presence of response groups, transcription unit correlation and differences in aggregated probe set intensities. Importantly, affiliated probe correlations and response groups are valuable diagnostic tools for detecting hidden factors.

Use cases were then employed to compare the outcomes when the modified gene values obtained with the ProbeSieve pipeline were used as input to the three most popular statistical methods (RMA, dChip and MAS 5.0). Since ArrayInitiative al-

lows us to create custom CDFs, we were able to explore the effects when the factor filters were combined with the statistical methods. Of principle interest was whether methods that improved gene-level correlation carried through to operon-level probe sets, since TRNs in E. coli should reflect TUs modulated by regulation of the operon.

## 4.2 Methods

### 4.2.1 Hardware and software

All of the data parsing, transformation and analysis scripts were written in Python 2.7 [105] and R 2.14.0 [114], and the ProbeSieve [104] database was developed using PostgreSQL 8.4, all under Ubuntu 10.04 LTS (Lucid Lynx). Individual Python and R packages used will be referenced where appropriate.

### 4.2.2 The E. coli antisense genome array

All of our datasets were produced on the GeneChip® E. coli Antisense Genome Array version 2 (Asv2). Manufactured by Affymetrix, this is a 3' in vitro transcription (IVT) expression array designed to report the abundance of 4,426 open reading frames (ORFs)/genes and 2,886 intergenic regions of the E. coli K12 genome (total number of probe sets = 7,312). Most probe sets have 14 or 15 probe pairs. Affymetrix originally designed the 141,629 probe pairs on the chip using version M54 of the E coli project that was made publically available through the project database housed at the University of Wisconsin/Madison (Blattner lab). Since the protocol for this array specifies that the mRNA (corresponding to the sense strand) be reverse transcribed into cDNA, the sequence of actual targets assayed to the array are identical to the antisense strand (hence the antisense in the name). Consequently, all of the complementary probes have the same sequence as the sense strand.

### 4.2.3 Data sets and databases

In this section, we discuss the data sets and databases used for this study.

ProbeSieve database

We created a relational database, ProbeSieve database, to store all of data related to this study, including data from third parties. It was developed using PostgresSQL [104]. Although we didn't use Data-FATE for this study, as discussed in Chapter 2 (Data-FATE), the Data-FATE development effort did inform the approach for developing methods to automate methods associated with the ProbeSieve database, such as defining database objects, loading data and partitioning data for the ProbeSieve database. We developed a consolidated DatabaseScripter class that automatically and consistently created the SQL for creating objects of the same type (e.g. multiple tables to store probe intensity data) and loading its corresponding data, similar to Data-FATE, although not as abstracted or automated. As is done in the Data-FATE framework, we partitioned, the especially large data sets, such as probe intensities, to reduce the time for loading and indexing. However, our partitions were not as fine-grained as suggested by the Data-FATE framework. For example, we stored the probe intensities for hundreds of experiments in a single table, rather than one experiment per table. We did this to avoid the cross-query problems discussed in the Data-FATE chapter.

RegulonDB

RegulonDB [115] is a comprehensive, curated and public database detailing the transcription regulatory network (TRN) of E. coli. It contains information about operons, transcription units, promoters, binding sites and terminators; much of this data is supported by experimental evidence, although some is based on computational predictions alone. We downloaded and installed a local copy of version 7.3 as a schema in the ProbeSieve database. We also added several views to make it easier to identify operon templates and single-template operons.

Faith data set

The 'Faith' data set consists of 507 *E. coli* Asv2 arrays that monitor E. coli transcriptome under "various conditions including pH changes, growth phases, antibiotics, heat shock, different media, varying oxygen concentrations, and numerous genetic perturbations." [40] This compendium consists of 241 publicly available, third-party arrays and 266 arrays that Faith et al. produced themselves (all of the CEL files are available in the Gene Expression Omnibus [44]). The purpose to making the compendium was to test the performance of the CLR algorithm, developed by Faith, against several competing network inference algorithms [116]. The probe set intensities for all of the arrays were uniformly normalized and summarized using RMA [83] and the derived data so produced was made publicly available from the M3D database [40]. We chose this data set because it has a large number of arrays and covers a wide range of perturbations arising from experimental conditions. In addition, having 266 arrays produced in the same lab, using the same experimental protocol and equipment minimizes a number of known sources of technical variation, useful as a control for the other datasets.

### 4.2.4 Re-mapping probes and calculating binding affinity

Before we could identify probes affected by individual physical and technical factors, we first needed to re-map the probes synthesized on the E. coli Asv2 array to a more current genome version and annotation, and then using the new mappings, calculate the binding affinity between all of the relevant probe-target pairs.

Re-mapping the probes to the genome

The goal of this step was to remap the probes on the E. coli Asv2 array to the most current version of the E. coli K12 genome sequence (from GenBank, NC_000913.2) and its annotation in RegulonDB (version 7.3) [115], finding both exact and inexact alignments. Finding exact alignments is the predominant – and relatively easy – method to re-map probes to their targets. However, we also needed to find inexact

alignments so that we could later add filters for non-ideal probe-target pairs to the ProbeSieve pipeline. Since mapping short microarray probe sequences (25nt) to a genome is the same problem as mapping HTS reads to a genome, we decided to use an HTS aligner. Although many aligners are available, such as Bowtie [117], MAQ [118] and SOAP [119], most of them only allow a few mismatches and don't allow gaps. However, we needed an aligner that could find alignments with at least six mismatches. This follows from the Kane criteria [78] for non-ideal binding affinity, which for 25mers only requires a core of 15 identical, consecutive bases (for minimum nucleation) and 75% sequence total sequence identity between a probe and its target to bind.

Based on these criteria, we chose to use the read aligner module of Mosaik [unpublished, [120]], which is a reference-guided assembler for next-gen sequencing data. Mosaik uses the Smith-Waterman algorithm [121] to align the probe to target, which allows for an arbitrary number of mismatches and gaps. However, Mosaik also allows you to specify a minimum 'core' of matches (corresponding to a minimum nucleation criterion) and the maximum number of mismatches allowed; these features made Mosaik an excellent fit for our alignment criteria.

We first built the target sequence (the entire E. coli K12 genome sequence) and the probe sequences (all of the perfect match and mismatch probes on the Affymetrix E. coli Asv2 array) using the MosaikBuild module. Using these sequence builds, we aligned the probe sequences to the K12 genome using the MosaikAligner module. Although we used most of the default options for MosaikAligner, we set the hash size to 15 (meaning that any matches must have at least 15 consecutive identities) and the maximum number of mismatches to 10. Gaps were not relevant for this study (all experiments were performed on the organism matching the probes, so the only gaps should be in the deletion mutants), so we set the gap open penalty and gap extension penalties to a high value (200), effectively preventing gaps in the

alignments. With these settings, MosaikAligner returned probe-genome alignments (in SAM format[122]) with at least 15 consecutive identities (minimum nucleation) and up to 10 mismatches percent identity $\geq 0.6$), slightly lower than the Kane value of 0.75, to include probes that are close to the cutoff.

We wrote Python scripts to parse and transform the SAM files into a more easily usable format. Using the alignment results, in conjuction with the gene annotations in RegulonDB [115], we then classified the probe-target alignments as belonging to a gene, an intergenic region or a junction between the first two categories. Specifically, to be classified as a gene alignment, the probe-target alignment must fall completely within the genomic boundaries of the gene; we similarly classified as an intergenic alignments. Probes aligning to multiple genomic entities were classified as junction alignments. For example, let's consider the two E. coli genes *thrL* (190-255) and *thrA* (337-2799), with an intergenic region defined between them (256-336). If a probe aligns to the genome at coordinates 195-219, it's a gene alignment; at coordinates 260-285, an intergenic alignment; at coordinates 325-349, a junction alignment. We stored all of this data in the ProbeSieve database.

Calculating the binding affinity between probe-target alignments

Using the exact and inexact aligments returned by Mosaik, we calculated the binding affinity between the probe-target pairs (where the probe targets a gene) using Oligonucleotide Modeling Platform™(OMP). Developed by DNA Software [123], OMP™is a commercial software application for "secondary structure analysis plus design and simulation of probes/primers, RT-PCR, Taqman, Multiplex PCR, LATE-PCR, Scorpions, Beacons, Allele Specific and FRET assays, Microarrays, RNAi, and new formats." [123] The microarray simulations were relevant to this study, as they simulate surface constraints that are imposed on the interactions of bound oligonucleotide probes with soluble targets. The limitation of the approach lies in its thermodynamic perspective, which assumes infinite time and no kinetic barriers to reaching

equilibrium. All thermodynamically favorable heteroduplexes are predicted to occur. We know this isn't always true: some probe-target pairs will never bind in a finite amount of time, while others don't have sufficient time to reach equilibrium. Currently, no application exists that takes into account the kinetics. We make a point of emphasizing in the Discussion that these results are likely to contain some false positive.

To calculate the probe-target binding affinities, we used the OMP Developer Edition, which includes a command-line interface to the OMP engine. To run an OMP simulation, you first need to create an experiment file (OEF), which defines the probes, targets and experimental conditions to use in the simulation. You can define a single probe and target, or you can include multiple probes and targets, for a multiplex simulation. In either case, you need to specify the probe and target sequences and the molecule type (RNA or DNA). For the experimental conditions, you are expected to specify: probe concentration, target concentration, assay temperature, various solution components (magensium, sodium, glycerol, DMSO, formamide, TMAC, and betaine), solution pH and whether or not polymer salt is present. Table 4.1 contains the hybridization conditions we used.

We didn't need to model competitive hybridization, since the probe concentration is significantly higher than target concentration (~ 10,000 fold difference), so we ran an individual simulation for each probe-target pair (~200,000). To automate the modeling process we developed a Python script to create an input file (OEF) for each simulation, run it, parse the output file (OOF) and merge the simulation results with an aggregate results, which we then uploaded to ProbeSieve DB. The Python script expects two input files: a file containing the hybridization conditions and a delimited text file containing each probe-target pair to simulate. The latter file requires a probe and target ID and their corresponding sequences (and the type of nucleic acid, DNA or RNA).

Table 4.1: Hybridization conditions for the OMP simulations.

| Parameter | Value |
|---|---|
| ASSAY_TEMPERATURE | 45.0 C |
| NUMANALY_MIN_TEMPERATURE | 0.0 C |
| NUMANALY_MAX_TEMPERATURE | 100.0 C |
| MAGNESIUM_CONCENTRATION | 0.0 M |
| SODIUM_CONCENTRATION | 0.5 M |
| GLYCEROL_CONCENTRATION | 0.0 M |
| DMSO_CONCENTRATION | 0.0 M |
| FORMAMIDE_CONCENTRATION | 0.0 M |
| TMAC_CONCENTRATION | 0.0 M |
| BETAINE_CONCENTRATION | 0.0 M |
| PH | 6.6 |
| POLYMER_SALT | TRUE |
| PROBE_CONCENTRATION | 1.86E-3 M |
| TARGET_CONCENTRATION | 3.0E-8 |

We created three different probe-target pair files for this study. From studies performed in out lab we knew that sometimes the folding of target DNA adjacent to the duplex-forming site alters the stability of the complex. The average length of the targets, as prepared using the Affymetrix protocol, is 50-75nt. In the first length file, we included the target sequence of exactly the length of the probe (length = 25); in the second, we extended the target sequence by one nucleotide on each end (length = 27); in the third, we extended the target by 38 on each end (length = 101). In total, we ran three sets of simulations, corresponding to each target length.

### 4.2.5 Individual factor filters

Having re-mapped probes and calculated probe-target binding affinities, we could identify probes affected by individual physical and technical factors. Here we discuss our approach for identifying affected probes for each of the factors, and then, how we correct for each of them.

#### Binding affinity between probe and target

The results of an OMP simulation return both a $\Delta$G and percent bound (PB) for each probe-target pair. If the value of $\Delta G \leq -10$ kcal/mol, we concluded that the

binding affinity between the pair was sufficient to maintain a significant signal after the washing step during an experiment. We collected the percent bound value, which tells you how much of the target is actually bound to probe at equilibrium, in order to modulate the reported intensity for that probe.

Probe-to-target remap

After calculating the binding affinity between both exact and inexact alignments, we knew which probe-target pairs were likely to bind. Given this, we identified probes that cross-hybridized (number of targets $\geq 1$), probes with an unidentified target (number of targets $= 0$) and misannotated probes (the probe was designed to measure on target, but after the remap, is now known to measure a different one). We deprecated any probe that cross-hybridized or had an unidentified target; we re-assigned any misannotated probe, but only if it had a single (although misannotated) target.

Probe sequence motifs

Using a specification file generated by ArrayInitiative, which contained the sequences for all of the probes on the E. coli Asv2 array we identified every probe with a G-run sequence motif ($\geq 4$ Gs in a row) and a list of probes containing the primer-spacer motif (CCTCC). If a probe had one of these motifs, we deprecated it.

Sensitivity range of the scanner

We employed two stages to account for limitations imposed by the sensitivity range of an Affymetrix scanner (noted in the technical specifications for the instrument), namely, that only reported intensities between 200 and 20,000 are reliable if comparing relative responses of targets in a sample. This has to be carried using the experimental data, and can vary between labs. Our goal was to select a subset of genes and operons containing probes with usable values across all of the experiments. In the first stage of the selection process, we examined groups of affiliated probes (e.g. genes, operons) and identified probes that were outliers. We counted the number of times an affiliated

probe's value fell within the linear range for each of the Faith arrays (values from 0-263), so that at the end, we had a count for each probe. We then determined which, if any, probes had a signficantly different counts from the remainder of the affiliated probes, using the interquartile range.. Intuitively, if the probe was an outlier, its response profile (in the linear response range of the scanner) would be inconsistent with other affiliated probes. Outlier probes were removed from the affiliated probe set.

In the second stage, we modified reported intensities directly by setting any raw intensities less than 200 to a value of 200, and likewise, any raw intensities above 20,000 to a value of 20,000. An unintended result of this thresholding was that the background correction module of the RMA algorithm was unable to perform correctly. Normalization in the set of tests using RMA was based only on the quantile normalization values.

#### 4.2.6 Combining individual filters into filter sets

After identifying and correcting for probes affected by individual factors, we combined them into a custom filter set, *SrSmLr*, which consists of the structure remap (based upon OMP binding affinities), sequence motif and linear range filters. We removed or re-assigned probes, and modified probe intensities, as appropriate to tge factor being tested, described above. The *Default* filter set was used as a control, based on the standard approach for background and affiliated probe set value estimation as applied by any of the analysis method. After defining these filter sets, we used ArrayInitiative to create custom array specifications.

#### 4.2.7 Creating the custom specifications

For each of the filter sets – *Default* and *SrSmLr* – we created several custom specifications. As stated above, for the *Default* filter set, we didn't remove or re-assign any of the probes but used the default specification, which included probe sets targeting intergenic regions and genes; we'll denote this specification as *Default-GiG*

Table 4.2: Array specifications.

| Specification name | Probe set grouping | Filters |
|---|---|---|
| Default-GiG | Gene and intergenic | None |
| Default-Gene | Gene | None |
| Default-Tu | Transcription unit | None |
| Default-Operon | Operon | None |
| SrSmLr-Gene | Gene | Structure remap + Sequence motif + Linear range |
| SrSmLr-Tu | Transcription unit | Structureremap + Sequence motif + Linear range |
| SrSmLr-Operon | Operon | Structureremap + Sequence motif + Linear range |

(GiG = gene + intergenic). We also created three new custom specifications which re-assigned probes to different groups. one such included only probe sets targeting a gene (*Default-Gene*), another that targeted transcription units (*Default-Tu*) and a third that target an entire operon (*Default-Operon*). We note that discriminating the second two correctly relies on annotation information and it may not always be correct. For the custom filter set, *SrSmLr*, we similarly created custom specifications for gene+intergenic, gene, transcription unit and operon, but included filters we wished to test. All of the specifications that we used for this study are summarized in Table 4.2. We used ArrayInitiative to create a CDF for each custom specification and made the corresponding Bioconductor CDF package using the *make.cdf.package* function in the *makecdfenv* package [111, 87].

### 4.2.8 Extracting probe intensities and summarizing probe sets

For each of the high-level specifications discussed in the previous section, *Default* and *SrSmLr*, we extracted probe intensities for each of arrays in the Faith (263) and Covert (43) data sets using Bioconductor's *affy* package [124, 87]. We only needed to extract the probe intensities once for each high-level specification; it was not necessary to repeat if for the sub-specifications (e.g. *Default-Gene*, *Default-Tu* etc.) because probe set groupings don't affect probe intensities, only probe set summarizations. For the *Default* set of specifications, we background-corrected and normalized the intensities in the same way as RMA. For the *SrSmLr* specification set, we only normalized

the probe intensities (again, conforming to the RMA method). As mentioned above, we did not background-correct because RMA's background correction was not able to carry out the function: when it was attempted it produced many null values. Once extracted and transformed, we then loaded those values into the ProbeSieve database.

For each of the sub-specifications discussed in the previous section, we summarized the probe set intensities using RMA [83], dChip [85] and MAS 5.0 [86], using the implementations in the *affy* Bioconductor package [124, 87]. We also developed Python code to summarize the probe set intensities using a standard mean and median. All summary values were loaded into the ProbeSieve database.

### 4.2.9 Calculating affiliated probe correlations

If the probes in a probe set are truly 'affiliated', that is if they're measuring the same target, then we expect that their measurements across many conditions will have a high, positive correlation. If not, then the probes are measuring different targets or groups of targets affected in different ways by the perturbation factors: in that case, depending on the influence of that probe, the probe set overall may perform poorly. To test for probes exhibiting this behavior, we calculated the correlation between the affiliated probes in each probe set, with categories as defined in Table 4.2: intergenic, gene, transcription unit and operon. Let's consider an example using a gene-level probe set. For each probe targeting the same gene (affiliated probes), we constructed an intensity profile using the Faith arrays (N = 263). Then, for every affiliated probe pair, we calculated the Spearman correlation between their profiles. As discussed in the Introduction, the more positively correlated the intensity profiles of two affiliated probes, the more likely it is that they are measuring target affected similarly by all perturbation factors. We calculated the affiliated probe correlation for every probe set defined in every array specification in Table 4.2 and stored the results in the ProbeSieve database.

### 4.2.10 Identifying response groups

For those probe sets with low affiliated probe correlation, can we find subsets of probes that have a higher intra-group correlation than inter-group correlation? Such subsets are called *response groups*. If a microarray were reporting intensities perfectly, every probe set is expected to have a single response group, with a highly positive average correlation. If a probe set has multiple response groups, and those groups have much higher intra-group than inter-group correlation, that probe set is actually measuring multiple factors or targets. This is the same concept as cross-hybridization. If a "probe" (in this case, a probe set) is measuring targets affected by multiple factors then in the measurement is likely to give a reading that is false for the indicated target,although it may occasionally be correct when factors balance out. The presence of multiple response groups is also a useful diagnostic tool - if you have multiple response groups in a probe set, there's likely a hidden factor that you haven't accounted for.

To identify response groups, we first constructed a weighted correlation network for each probe set (in each specification), where each node is a perfect match probe and an edge between them represents their correlation. Since this is a weighted network, every pair of nodes connects with an edge, with the Spearman correlation value as the edge weight. We then ran the weighted version of the fast greedy optimization [125] of the Newman algorithm [126], as implemented in the Python implemetation of *igraph* library [127], to find communities in the graph. A community (or module) is a subnetwork of nodes that have denser internal connections than with other nodes in the network [128]. Since these are not cliques, the communities are allowed to overlap, but the connections between them will be sparser. Since the edge weight in our graphs is the correlation between probe intensity profiles, a community (which we're calling a response group) corresponds to nodes that have a higher correlation with each other than with other probes in the probe set. So, if the intra-group correlation

has a high enough positive correlation, then the probes in that response group are likely measuring the same factor or target. If the inter-group correlation between two response groups is low enough, and assuming high intra-group correlations, then we can say that the two response groups are sensitive to different factors or targets.

There are two major aspects of community detection algorithms which will affect our interpretation of response groups. These algorithms work by finding partitions of the graph (subnetworks) such that the connections between group members are denser than those between groups. However, they need only be *slightly* denser for the algorithm to partition the graph; in the parlance of response groups, the intra-group average correlation need only be slightly greater than the inter-group average correlation. For example, let's assume that a probe set has two response groups, A and B, and further, group A's intra-group average correlation is 0.8, group B's intra-group correlation is 0.82 and the inter-group corrrelation is 0.75. A community detection algorithm will always identify two different response groups because that is the optimal partition. However, we can easily make the case that the inter-group correlation is sufficiently high to merge them back into a single response group. A natural question is, 'How low must the inter-group correlation be, as compared to the intra-group correlations, to keep them as separate groups? '. There's also no guarantee that the intra-group correlations will be high - this is the second problematic aspect of the algorithms. For example, let's again assume we have response groups A and B, but this time, group A's intra-group average correlation is 0.2, group B's intra-group correlation is 0.15 and the inter-group corrrelation is 0.1. Again, the community detection algorithm will identify the two groups, and even if we can make the case for not merging them, the affiliated probes in each group have such a low correlation that they're likely responding to different factors or targets. Considering these two scenarios, we need to develop a method to refine the groups by merging them when appropriate.

#### 4.2.11 Evaluation and diagnostic methods

As discussed in the Introduction, when researchers develop either statistical or factor-based methods to improve their interpretation of microarray measurements, they tend to evaluate the impact of their changes using an idiosyncratic set of evaluation methods. To comprehensively evaluate a method, it's best to use multiple tests, including comparisons to a well-regarded standard approach and relative comparisons of that standard to proposed novel approaches. For this chapter, we've developed novel techniques and used published approaches to evalate the impact of our filters. Here we discuss the evaluation methods that we've implemented so far.

Affiliated probe correlation

If a probe set faithfully measures its target, we expect that its affiliated probes will have a high positive correlation. Hence, if we improve a probe set's definition, we expect to see an increase in positive correlation, which, when applied to all of the probe sets whose definition have changed post-filtering, can be used to evaluate the impact of the filter set. If the filter set, on average, improved our measurements, we expect to see an average increase; if the filter made the measurements worse, we expect to see a decrease in average affiliated probe correlations. We look at changes in affiliated probe correlation for gene-, transcription unit- and operon-level probe sets by visual inspection of a density plot.

Transcription unit correlation

The operons in *E. coli* (2,292 of them, according to RegulonDB) produce single polycistronic mRNAs, which are translated to produce the multiple gene products. Belonging to a single transcription unit, the probe sets measuring each of its component genes are actually measuring the same transcript. By definition, these probe sets are a type of affiliated probe, and thus, we expect that they will have highly correlated expression profiles. Put another way, we expect the genes in an operon to be positively correlated. This has been used as an evaluation tool by Harr [129] and

Alvarez [130]; the former for testing the impact of normalization methods, the latter for testing the impact of re-mapping probes to the genome.

However, this expectation is only correct when there is a single possible transcription unit (TU). A single operon can produce multiple transcripts (using alternative transcription start sites [131]. For example, let's assume we have an operon composed of genes A, B and C, in that order. By definition, this operon must always produce the transcript ABC; however, it can also produce the transcripts AB, BC, A, B or C. Each of these are templates for an alternative transcription unit. We say they're templates, because the same template (e.g. AB) can be produced by multiple promoters. Technically, the combination of template plus promoter is an alternative transcription unit, but expression microarrays cannot discriminate promoter differences . When an operon produces only the single, canonical transcript (e.g. ABC in the example), then we say that the operon is a *single-template operon*. Since some operons can produce different transcripts, we only expect the genes in the same transcription unit, not necessarily the same operon (unless it's in a single-template operon), to always have a high positive correlation.

As with affiliated probe correlations, of which this is a special case, we expect that if we improve our specification post-filtering, we'll see an increase in positive correlation between genes in the same transcription unit, assuming the probe set definitions were significantly altered. This would allow us to detect the presence of multiple TUs from an operon.

Affiliated probe correlation vs. transcription unit correlation

We expect that as we increase the affiliated probe correlations at the gene-level we should also see a corresponding increase in transcription unit correlations. That is, we expect these two levels of correlation to be correlated. As we improve affiliated probe correlations, we should also improve transcription unit correlations. Any significant deviation from this behavior suggests the influence of a hidden factor. This is serves

as both an evaluation method and a diagnostic method.

Differences in summarized probe set intensities

Ultimately, summarized probe set intensities are used as the proxy measure for relative target concentration, and any significant changes to them can change the results of downstream analyses, such as detecting differential expression or inferring networks. So here, as we did in the ArrayInitiative case study, we determined how, on average, the expression values generated using the custom specifications changed with respect to the default expression values.

For a single probe set on a single array, we calculated the expression difference as follows:

$$\Delta = \frac{E_c - E_d}{E_d} * 100$$

where $E_c$ is the custom expression value and $E_d$ is the default expression value. We calculated the expression differences for every combination of probeset (number depends upon specification), Faith array (263) and summarization method (RMA, dChip, MAS 5.0, mean, median).

There is one important difference between this evaluation method and the one used for the ArrayInitiative case study. In ArrayInitiative, we only removed probes when creating the custom specifications, and consequently, we calculated the average difference (delta), across all of the RAND arrays, as we removed probe pairs. In this study, since we both removed and re-assigned probes, we calculated the average delta, across all of the Faith arrays, as we introduced changes to the probe set definitions (where a change is defined as either adding or removing a probe).

## 4.3 Results and discusssion

### 4.3.1 Changes to probe set definitions

Each component of the *SrSmLr* filter set identified probes affected by a specific physical or technical factor, which were then removed or re-assigned, as appropri-

Table 4.3: The number (and percentange) of probes affected by specific factors.

| Factor | Probe Pairs Affected |
|---|---|
| Unidentifiable target (exact) | 1,974 (1.4%) |
| Cross-hybridization (exact) | 2,089 (1.5%) |
| Unidentifiable target (structure) | 1,428 (1.0%) |
| Cross-hybridization (structure) | 27,070 (19%) |
| G-runs | 4,663 (3.3%) |
| Primer spacers | 1,020 (0.7%) |
| Linear range | 5,305 (3.7%) |

Table 4.4: Number of probe sets and probe pairs (in parentheses) in different specifications.

| Specification | Genes + intergenic | Genes | Transcription unit | Operon |
|---|---|---|---|---|
| Default | 7,312 (141,629) | 3,867 (59,577) | 2,773 (70,777) | 2,292 (59,051) |
| SrSmLr | N/A | 4,292 (52,327) | 3,023 (62,011) | 2,499 (51,832) |

ate, in the custom specifications. Table 4.3 shows the number of probes that were removed because they were affected by a particular factor, such as an unidentifiable target, cross-hybridization, sequence motifs and linear range. For the unidentifiable target and cross-hybridization factors, we show how many probes were categorized that way when using an exact alignment and a structural alignment (based upon the OMP results). By including the structural alignments (which turned out to be largely equivalent; discussed below), we removed fewer probes due to an unidentifiable target, but we removed significantly more probes because of cross-hybridization. Since coverage of the target space is important for characterizing networks, we were concerned that removing ~30% of the probe pairs would impact the number of genes and transcription units that we could monitor. However, the structural remap turned out to increase the coverage of the target space, as can be seen in Table 4.4. We increased the gene coverage by 11%, increased the transcription unit coverage by 9% and the operon coverage by 9%. Another important question to answer was how many, and to what extent, were the probe set definitions changed after filtering. Figure 4.1 shows the percentage changes to the definition of the common probe sets between

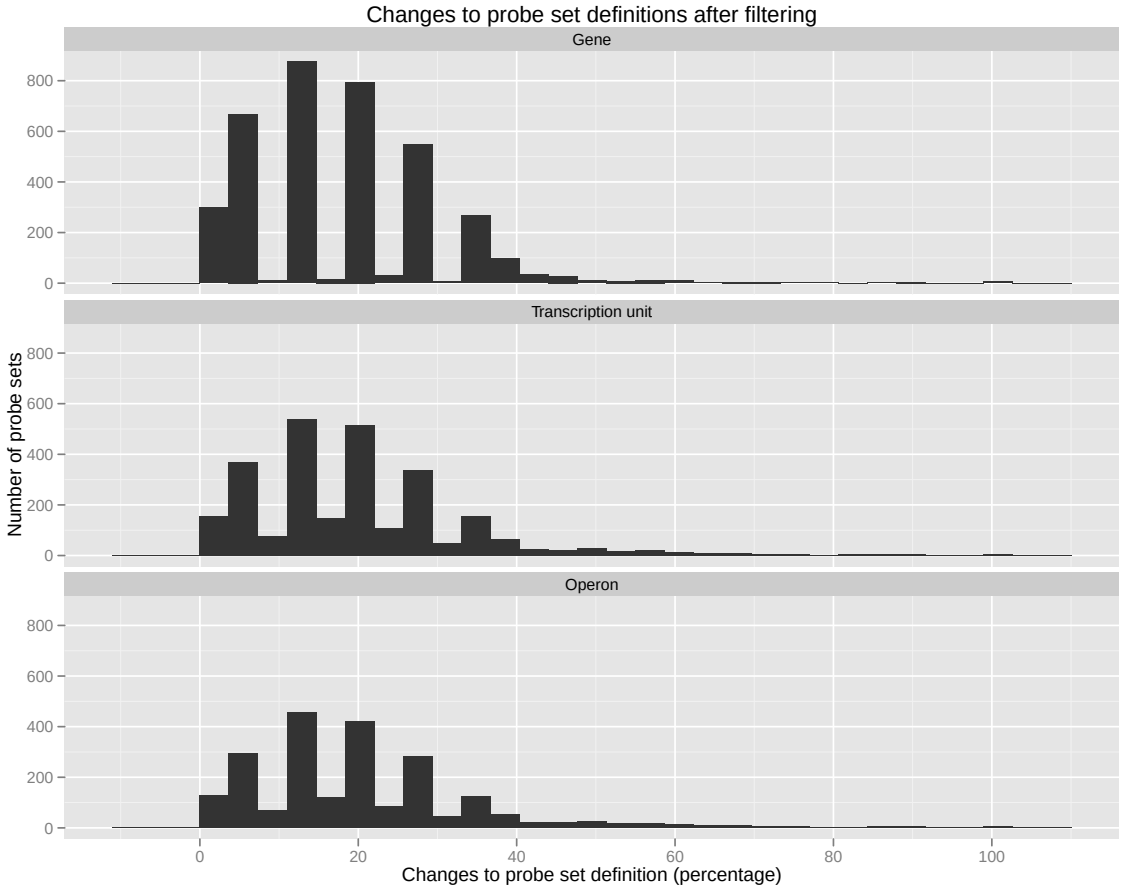


Figure 4.1: The number of changes to probe set definitions after filter (specification = SrSmLr).

the *Default* and *SrSmLr* after filtering. A change (or difference) is defined as the addition or subtraction of a probe pair to the probe set in the *SrSmLr* specification, as compared to the *Default* specification. We find that a significant number of probe set definitions have between 10-40% of their probe makeup changed after filtering for each of the sub-specifications, SrSmLr-Gene, SrSmLr-Tu and SrSmLr-Operon.

### 4.3.2 Binding affinity between probes and targets

We planned to use the binding affinity between probes and targets, as reported by the OMP simulations, to modulate probe intensities to better estimate the transcript abundance. For example, knowing a probe-target pair only had a binding affinity of 60% (that is, only 60% of the available target would likely bind), we could calibrate

the original intensity using this information. However, we found that, for all OMP simulations, for almost every probe-target pair that met the Kane criteria the binding efficiency was 100%. Consequently, this value provided no basis for modulating the intensities. However, we suspect that these simulations were not accurate, as discussed later.

### 4.3.3 Affiliated probe correlations

After determining that the *SrSmLr* filter set makes substantial changes to a large number of probe sets, we wanted to see if the filters improved the affiliated probe correlation. We compared the distribution of all of the affiliated probe correlations from the *Default* specifications to the *SrSmLr* specifications, shown in Figure 4.2. At all levels, gene, transcription unit and operon, the affiliated probe correlation decreases. Is this also true for probe sets whose definitions have significantly changed? In Figure 4.3, we see that, for probe sets have 50% or less probe pairs in common between the *Default* and *SrSmLr* specifications, the filtering improved for the affiliated probe correlation for all levels, especially for genes.

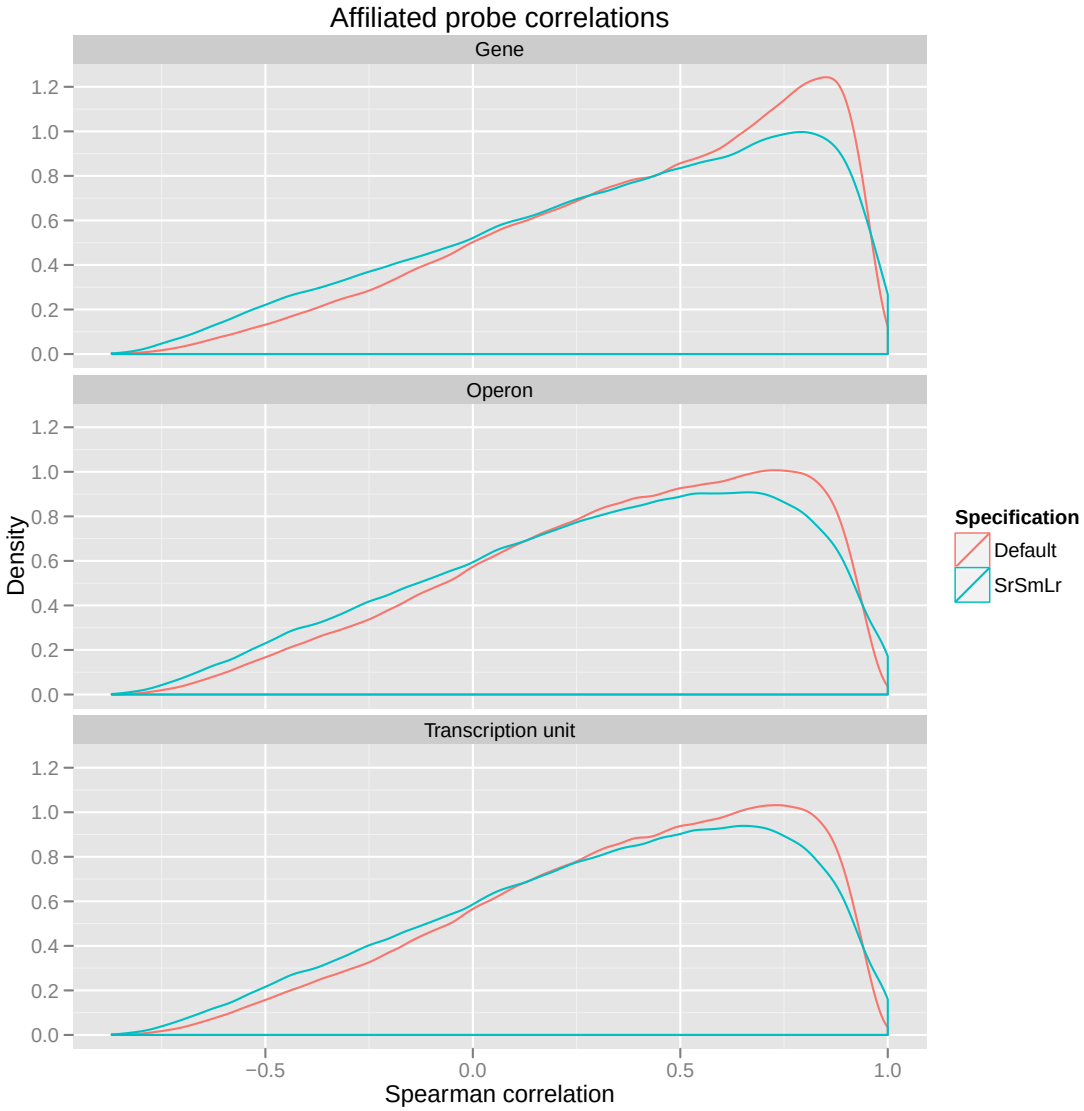


Figure 4.2: Comparison of affiliated probe correlations. All probe sets.

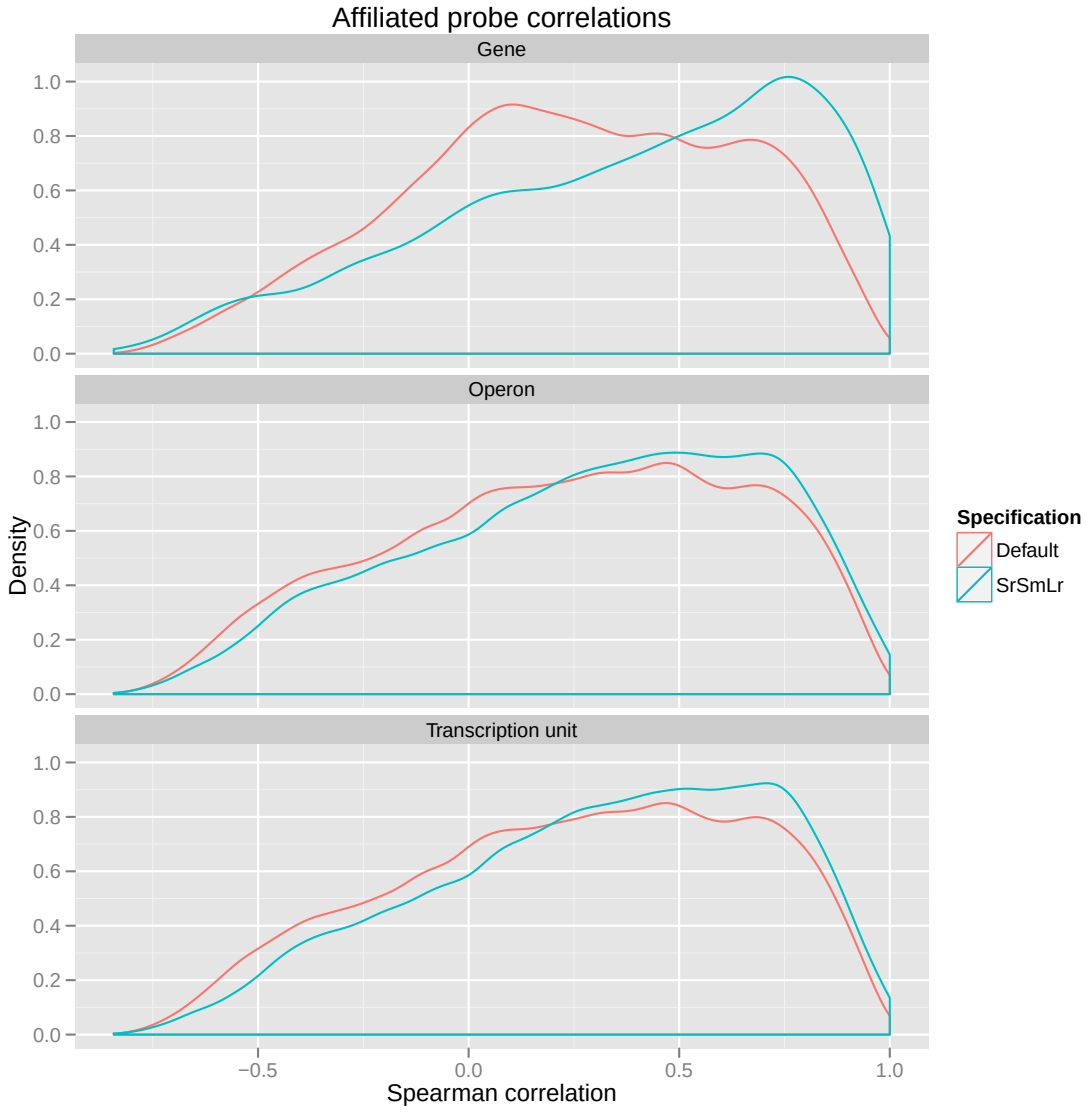


Figure 4.3: Comparison of affiliated probe correlations. Only probe sets with a 50% change.

Does this mean that filtering made the microarray measurements worse? Or did filtering remove a systematic bias that was artificially boosting the positive correlation seen in the *Default* specifications? The latter scenario is likely, given that cross-hybridization, sequence motifs and values below the linear range are known to induce spurious, positive correlations, and that a significant number of probes were affected by these factors. By removing the affected probes, we likely removed posi-

tive correlation bias that didn't reflect the actual correlations, thus giving us a more accurate, but less correlated set of measurements.

Regardless of the answer, Figure 4.2 clearly shows a significant portion of the affiliated probe correlations, in both the *Default* and *SrSmLr* specifications, are less than 0.5. Moreover, a substantial portion of the correlations are actually negative. With or without filtering, poor affiliated probe correlation seems to be standard, suggesting that there are other factors affecting the probes have not been accounted for.

### 4.3.4 Response groups

Since factor-based methods did not give rise to the expected improvements, another approach is to determine whether subsets of the probes in an affiliated group are better correlated. The first step is to establish a baseline: how prevalent are response groups inthe *Default* specification? Do we improve the situation (reducing their number) by filtering? Figure 4.4 shows the number of response groups in probe sets in the Default and SrSmLr specifications. We immediately see that very few probe sets in the Default specifications have a single response group, which is the desired number. Most probe sets have either two or three response groups, meaning that some or all of the probes in probe sets is likely responding to multiple factors or targets; the problem is to determine which best reflects the true TU concentration. These results must be taken with a bit of caution, however. Figure 4.4 only shows the number of response groups, not the associated affiliated probe correlation. A probe set might have multiple response groups, but this is not meaningful if the inter-group correlation is almost as high as the inter-group correlation, suggesting that they are indeed a single response group and should be merged. To allay these concerns, Figure 4.5 shows the distribution of correlations within and between groups. From these distributions, we can see that the correlation within response groups is significantly higher than between groups, as hoped. However, there is still significant overlap between the

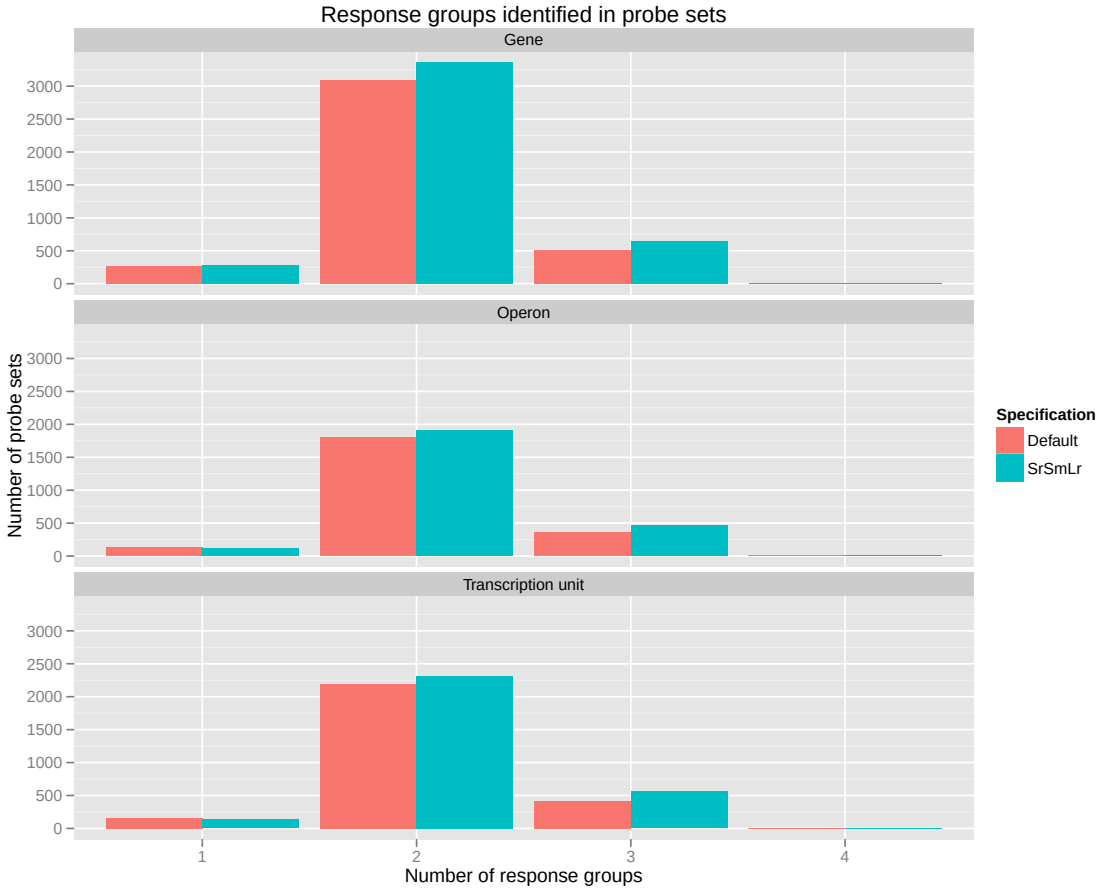


Figure 4.4: Prevalence of response groups in each specification.

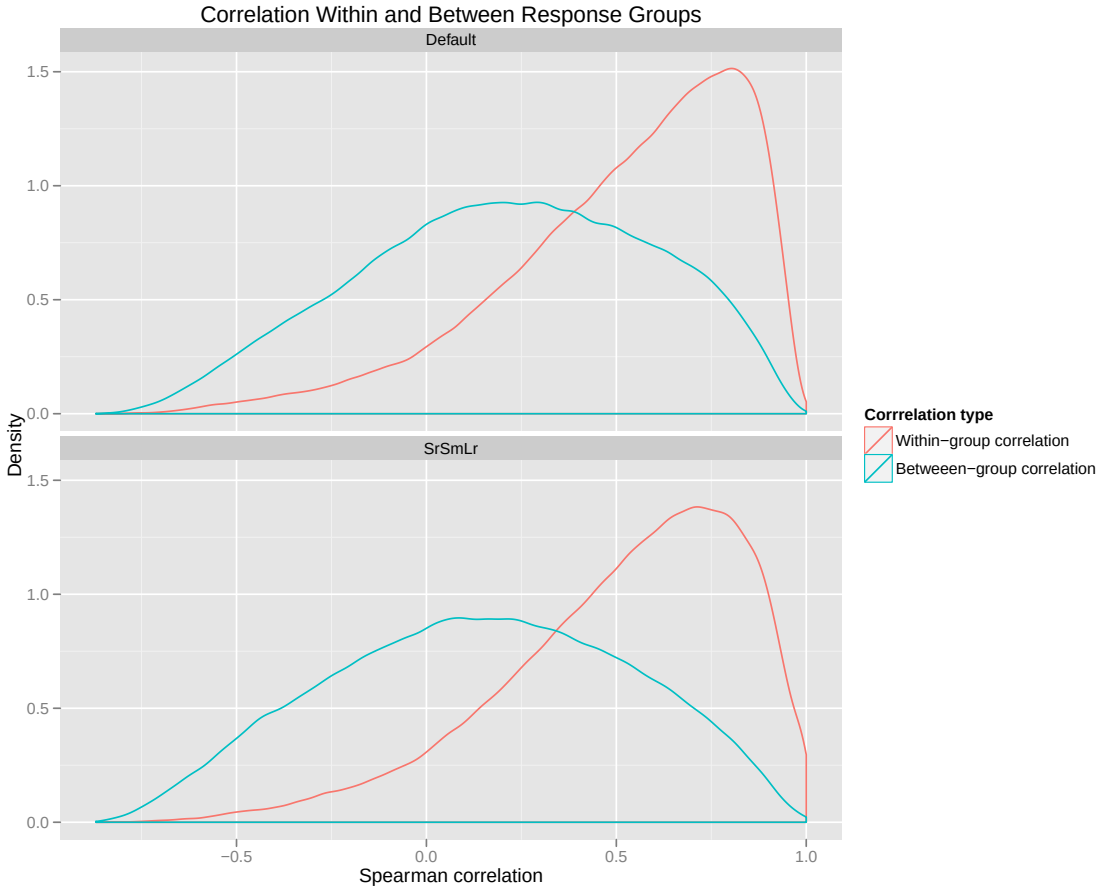


Figure 4.5: Distribution of correlation within and between response groups.

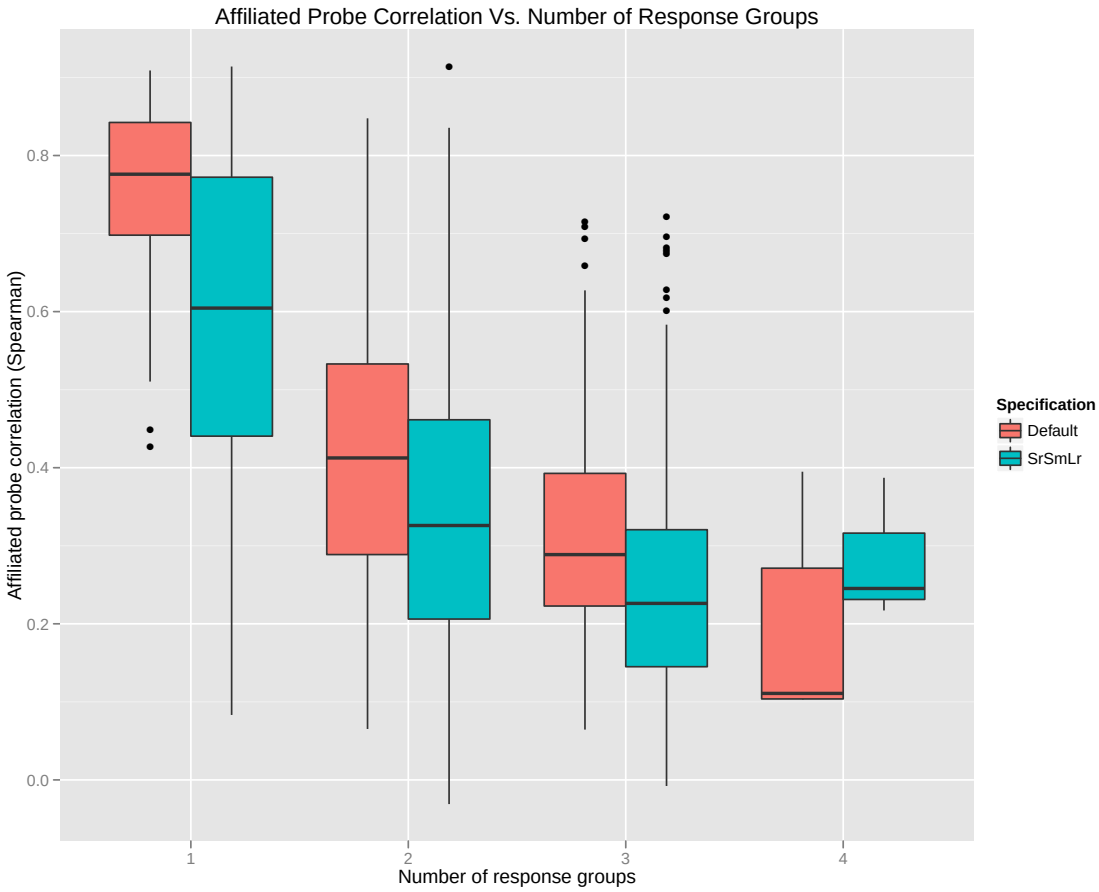


Figure 4.6: Response groups vs. affiliated probe correlation.

two, indicating that we need to develop criteria for merging response groups when the inter-group correlations are not much lower than the intra-group correlations.

The presence of multiple response groups in the majority of probe sets suggests that there is one or more hidden factors still affecting the probes on this array. Moreover, since they sort into groups, it's likely not a random (noise) factor.

### 4.3.5 Response groups vs. affiliated probe correlation

Here we investigated the relationship between response groups and affiliated probe correlations. Do probe sets with less response groups tend to have a better average affiliated probe correlation? Figure 4.6 shows this relationship. We see that indeed, probe sets with fewer response groups tend to have a higher average affiliated probe correlation. This suggests that reducing the number of response groups should show

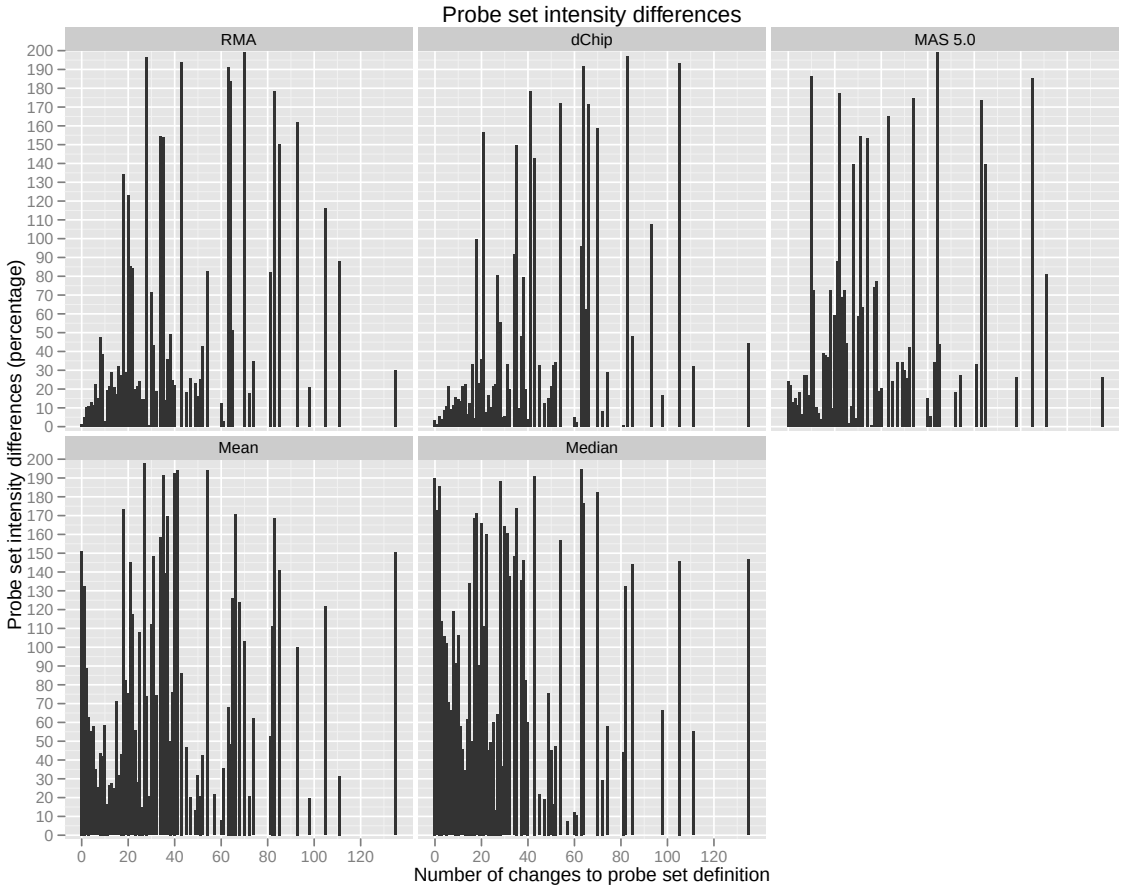


Figure 4.7: Differences in summarized probe set intensities, for transcription units, after filtering.

an increase in the affiliated probe correlation. As we saw when looking at just the affiliated probe correlations, the affiliated probe correlations for the *Default* specification are generally higher than those in the *SrSmLr* specification.

### 4.3.6 Differences in summarized probe set intensities

Are there significant changes to expression estimates in the custom specifications compared to the default specification? If so, does this occur as a function of the number of changes made to a probe set? For each of the five summarization methods, we calculated the percent different between probe set intensites in the *SrSmLr* specifications as compared to the *Default* specifications. Fig. 4.7 shows the results of this analysis for transcription units. As we can see, the probe set intensities post-filtering

were significantly different. For the mean and median summarization methods, the probe set intensities are significantly different post-filtering even when the number of changes to the probe set definition are minimal. This is due to the second stage of linear range adjustment – where we set values less than 200 to 200 and any values greater than 20,000 to 20,000 – which we only could use for the mean and median summarization methods. Although the observed differences do not necessarily mean that filtered probe set intensities are more accurate, they are likely to alter our interpretation of microarray experiments, such as when identifying differentially expressed genes or inferring TRNs from microarray data for this platform (at least for certain genes).

### 4.3.7 Transcription unit correlation

Here we assess the impact of the *SrSmLr* filter on the correlation of genes in a transcription unit. Namely, do the genes that make up a transcription unit become more or less correlated after filtering? The values used to build the expression profiles for each gene were summarized using RMA, dChip, MAS 5.0, with both mean and median as summarization method. Using these profiles, we calculated the correlation between the genes in each transcription unit. Figure 4.8 compares the distribution of gene correlations in the *Default* and *SrSmLr* specifications, grouped by summarization method. . For all summarization methods, the transcription unit correlation decreases after applying the filters. Is this also true for transcription units where the probe set definitions for its component genes have changed? Figure 4.9 is the same as Figure 4.8, except here we required that the definition of at least one probe set had changed for a transcription unit to be included. We see the same trend as before - the transcription unit correlation decreases after applying the filters. This is perhaps not surprising, given that the affiliated probe correlations decreased after filtering, and that affiliated probe correlations are positively associated with transcription unit correlation shown in the next section.

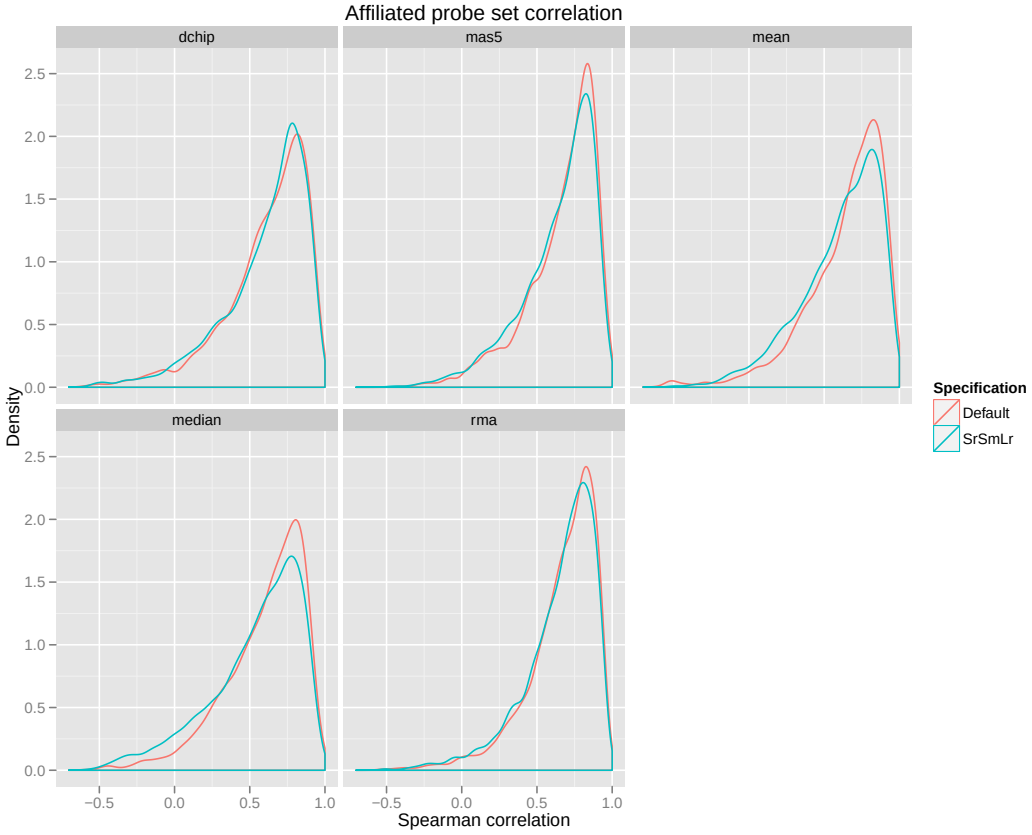


Figure 4.8: Correlation for all transcription units.

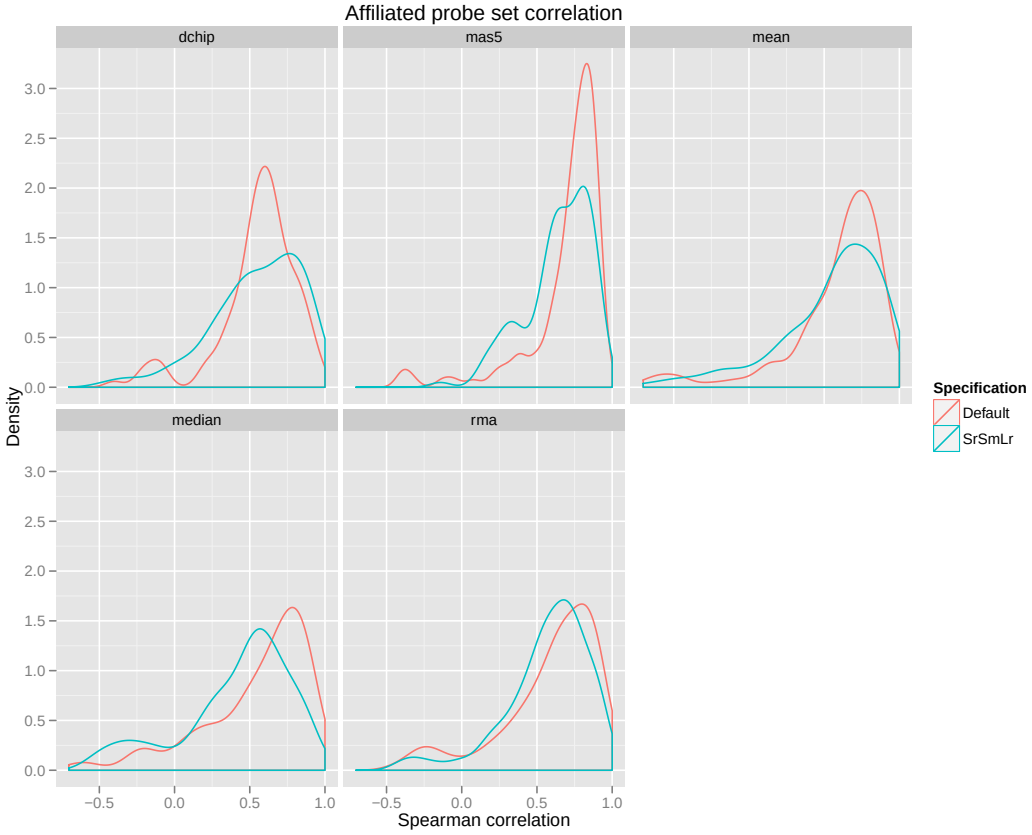


Figure 4.9: Correlation for all changed transcription units.

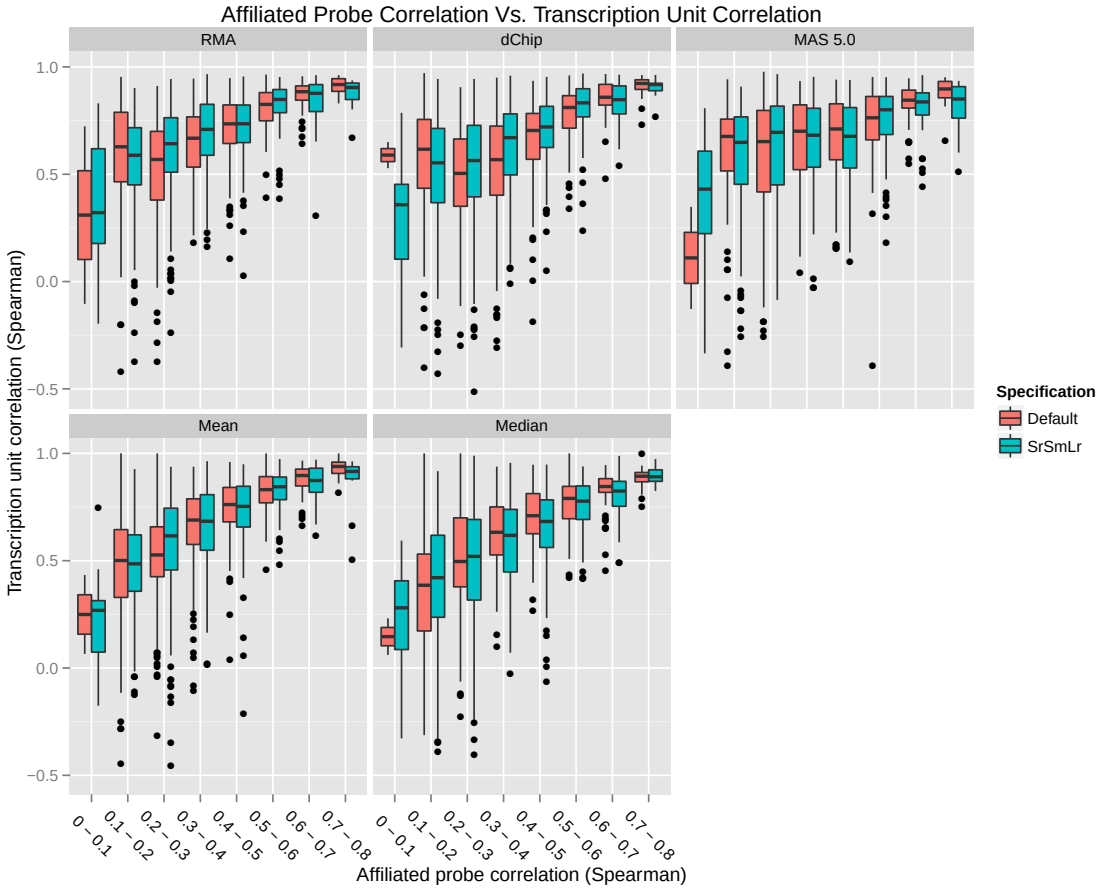


Figure 4.10: Affiliated probe correlation vs. transcription unit correlation.

### 4.3.8 Affiliated probe correlation vs. transcription unit correlation

Here we investigated the relationship between affiliated probe correlations and transcription unit correlations. Specifically, we wanted to find out if improving the former tended to improve the latter. Figure 4.10 shows this relationship. We see that there is a positive relationship between affiliated probe correlation and its corresponding transcription unit correlation. While the positive trend exists, the correlation is fairly low (need to calculate this value). This trend holds for both the *Default* and *SrSmLr* specifications.

These results suggest several things. First, by improving affiliated probe correlations, you'll tend to, on average, also improve the transcription unit correlation.

However, since the correlation between these two variables is low, there are other factors affecting transcription unit correlation, besides affiliated probe correlation. Finally, we expect that if our filtering significantly improved the measurements, we might still see a positive trend, but it would be compressed into a much narrower band of highly positive correlations. Since we see the positive trend across a much broader range (0 - 0.8), this again suggests that additional hidden factors are affecting the probe responses.

## 4.4 Discussion and future work

Our plans for future work fall into four broad categories: response groups, binding affinity between probe and targets, investigation of hidden factors and improvements and extensions to the ProbeSieve pipeline.

### Response groups

Although we showed that probes in the same response group have, on average, significantly higher correlations than probes not in the same group, there is still overlap, and in some cases, two response groups should probably be re-merged into a single group. To do this, we'll need to extend our methodology for identifying response groups to refine the initial partitions by the community detection algorithm. The unresolved question is, 'How much separation between two groups do we need to consider them two distinct groups that are measuring different factors or targets?'. In this case experiments that measure gene levels with a measure such as PCR might provide guidance, but there are very few such datasets with more than one or two PCR assays accompanying a microarray assay, on the same genotype and conditions as those used in our experiments.

Having applied the refinement method and identified the final set of response groups for each probe set, we want to compare the response groups in the *Default* specifications with the filtered specifications ($SrSmLr$). Were certain response groups in the *Default* specification enriched for certain factors? If so, we should see that

the filters removed whole response groups. How many response groups are common between the *Default* and filtered specifications? How many new ones are there? Are any of the response groups similar to each other, and if so, does this correspond to a certain, hidden factor?

Finally, we've seen that by reducing the number of response groups, we tend to get improved transcription unit correlation. Again, we still need to method for identifying the group that reflects the true concentration. At the moment we are searching for network or biological features of the response groups that can guide our choice.

Binding affinity between probes and targets

When determining if two probes will bind to the same target, we'd like to make the criteria even less stringent than suggested by the Kane criteria. Studies in our lab and others [76] have suggested that the minimum required nucleation and percent identity are actually much lower. Even using the Kane criteria, in concert with binding affinity data from OMP, we identified and removed many more cross-hybridizing probes than when using an exact alignment. By further relaxing the alignment criteria, we may identify additional cross-hybridizing probes whose removal will improve the array measurements, leading to fewer response groups and, improved affiliated probe correlation. Finally, it may be that the kinetic information for the extent of reaction for each duplex is the unresolved set of factors. We have not identified a kinetic simulator that can handle the complexity of the target mixture in microarray experiments. Such a simulator must be able to handle competition between affiliated probes for the same target. Since the average length of the fragmented target is 100 using the protocol recommended for the E. coli antisense genome array, any probes that bind to the same target within 50-100 nucleotides of each other might very well compete for the same fragment. Thus simulations could limited to competition between probes that bind to regions of the target that are that close.

Existing and hidden factors

We first plan to determine the impact of individal factors that we know about (or combinations of them). We also plan to propose new factors that might be affecting the probes on the array, as suggested by the presence of so many response groups, and then test them using the pipeline. Here we might also use factor analysis to identify new factors and determine the relative importance of individual factors.

Improvements and extensions to the pipeline

To improve and extend the ProbeSieve pipeline, we'd first like to add more evaluation methods, such as concordance of lists of differentially expressed genes, correlation between independent, replicate sample, correlation with RT-PCR results and regulon correlation, to name a few. Once the steps in the pipeline are largely finalized, we'd like to apply it to other prokaryotes, such as Salmonella for which many datasets exist and, based upon our experiences, develop an API and GUI so that others can easily extend the process to their favorite prokaryotes. Also, we'd like to develop a method to rate the quality of an array platform, and individual probe sets, based the results of the evaluation methods, especially affiliated probe correlations and response groups.

# CHAPTER 5: CONCLUSIONS

The main theme of this dissertation was to improve methods for storing, managing, modeling and retrieving data (Chapter 2 - Data-FATE) and to develop tools and methods for improving our interpretation of Affymetrix microarray data (Chapter 3 - ArrayIniative). These are essential supporting tasks for systems biology research, because we need complete and reliable microarray measurements to effectively characterize a transcription regulatory network, which is the most fundamental network in a biological system.

## 5.1 Data-FATE

We developed Data-FATE for those researchers who want to develop persistent descriptive ontologies for their experiments, without needing to implement their own ontological layer. The ontological model extends the relational model by integrating parts of an ontology – concepts and attributes – with the definition of relational objects. The SIMS extends the RDBMS to implement the ontological model, providing tools to define and curate ontological data types, associate data with QT sets and to automatically create and load relational tables based on the associate QT set. The SIMS also improves performance by partitioning data associated with the same QT set into different tables. Whenever researchers store data in a relational database, they are constructing a descriptive ontology. The Data-FATE framework– ontological data model and SIMS – formalizes this, allowing for more consistent and unambiguous representations of scientific data and maintaining the simplicity and power of prevalent relational databases.

While our re-design to the Data-FATE SIMS significantly improved it, we didn't

end up using it for the ArrayInitiative or ProbeSieve studies because it lacked support for cross-table queries. Developing this feature, and others discussed in the DataFATE chapter, would have required a significant amount of additional resources (time and money). Since the ArrayInitiative and ProbeSieve studies used hundreds of microarrays for the analysis, we concluded that, without cross-table queries, keeping track of and writing queries against hundreds of tables (corresponding to the array data) would be counterproductive. Still, we used the general philosophy behind DataFATE – automatic creation of scripts to create database objects, consistency of data types and partioning of data (to a moderate extent) – when we developed the scripts to create database objects for and load into the ProbeSieve database.

## 5.2 ArrayInitiative

ArrayInitiative is a software tool designed for those biological researchers who want to create custom microarray specifications, such as a CDF, without the additional burden of learning the manufacturer's specification file format or learning an API. It provides graphical tools for importing a manufactuer's microarray specification, defining custom versions of a manufacturer's specification, writing array specifications in their standard format or in an easily understandable, non-standard representation. Creating a custom array specification requires only minimal knowledge of a manufacturer's specification standards (file formats and logical rules) and the ability to create a simple delimited or XML file.

he case study illustrated two concepts: the simplicity of using ArrayInitiative to create custom array specifications and how those modified specifications can significantly change summarized expression values. By not being constrained to a specific strategy for re-defining an array specification, ArrayInitiative enables researchers to create new specifications based upon their own requirements. These array specifications might be the result of a new probe-filtering technique or may help to answer a specific biological question using a unique set of probes to define a gene. Since it is

unclear which re-definition strategies are the best, ArrayInitiative will make it easier to rapidly define and test competing approaches and compare them to the manufacturer's array specification. By retaining the expected format of the cdf file, researcher can use established, tested software such as RMA.

## 5.3 ProbeSieve

In this chapter, we applied a modified version of the BaFL pipeline to data sets acquired using the E. coli antisense genome array (version 2). We extended the set of filters by adding a sequence motif filter and updated the re-mapping filters to allow for inexact alignments (meeting criteria even less stringent than the Kane criteria) that are likely to bind base upon binding affinities returned by OMP. We showed that the probe set definitions were significantly and uniformly changed post-filtering, especially by accounting for cross-hybridization from inexact alignments with a high binding affinity. Despite removing a large number of probes, we actually increased the number of genes and transcription units covered by the array, allowing better coverage of the genome when modeling E. coli transcription regulatory networks.

We then evaluated the impact of the filter set using a number of novel and published methods. The assumption tested was that removal of probes known to be subject to multiple factors would improve the correlation of the remaining probes in a designated set. As sets we used the original gene group minus the problem probes, and subsets of that group identified as response groups; we also tested probe sets defined by membership in transcription units in multi-gene operons. We found that the affiliated probe correlation and transcription unit correlation decreases upon the removal of multi-factor response probes. By removing probes affected by cross-hybridization, sequence motifs and linear range sensitivity, we removed spurious positive correlations that were introducing a positive bias into unfiltered affiliated probe correlations. Transcription unit patterns corresponded to the patterns of the component genes, that is, decreasing affiliated probe correlations correspond to decreasing

transcription unit correlations (and vice versa). However, as we saw with the ArrayInitiative study, the filters induced a significant change in the summarized probe set intensities, which will likely have an effect on downstream analyses, such as identifying differentially expressed genes or inferring TRNs.

We also introduced the concept of a response group, showing that most probe sets in the Default specifications separate into multiple response groups, and showed that when multiple response groups occur they correspond to lower average affiliated probe correlations. We also showed that the majority of probe sets also contained multiple response groups post-filtering, suggesting that there are remaining factors affecting the probes on the E. coli antisense array.

## 5.4 Summary

In Chapter 2, we significantly improved the prototype version of the Data-FATE SIMS, which was and is used by several members of the Weller lab (`http://webpages.uncc.edu/~dcarr10/DataFATE_html/DataFATE_Home.html`). In Chapter 3, we developed ArrayInitiative, which provides the research community with a valuable tool for simplifying the process of creating custom CDFs (and was used extensively in the Chapter 4). ArrayInitiative was published in BMC Bioinformatics, 2011 (`http://www.biomedcentral.com/1471-2105/12/136/`). In Chapter 4, we adapted and extended the BaFL pipeline to the E. coli antisense genome array version 2. We introduced the concepts of affiliated probes and response groups, and using both as a diagnostic measure, showed that, even after using standard and upgraded filters, the array has one or more hidden factors affecting its probes, and thus, confounding our interpretation of microarray experiments using this platform.

# REFERENCES


[1] Kitano H: Computational systems biology. *Nature* 2002, 420(6912):206–210, [http://dx.doi.org/10.1038/nature01254].

[2] Kitano H: Systems biology: a brief overview. *Science (New York, N.Y.)* 2002, 295(5560):1662–1664, [http://dx.doi.org/10.1126/science.1069492].

[3] Purnick PEM, Weiss R: The second wave of synthetic biology: from modules to systems. *Nat Rev Mol Cell Biol* 2009, 10(6):410–422, [http://dx.doi.org/10.1038/nrm2698].

[4] Pinheiro VB, Taylor AI, Cozens C, Abramov M, Renders M, Zhang S, Chaput JC, Wengel J, Peak-Chew SY, McLaughlin SH, Herdewijn P, Holliger P: Synthetic Genetic Polymers Capable of Heredity and Evolution. *Science* 2012, 336(6079):341–344, [http://dx.doi.org/10.1126/science.1217622].

[5] Pavlopoulos GA, Secrier M, Moschopoulos CN, Soldatos TG, Kossida S, Aerts J, Schneider R, Bagos PG: Using graph theory to analyze biological networks. *BioData mining* 2011, 4:10+, [http://dx.doi.org/10.1186/1756-0381-4-10].

[6] Giot L, Bader JS, Brouwer C, Chaudhuri A, Kuang B, Li Y, Hao YL, Ooi CE, Godwin B, Vitols E, Vijayadamodar G, Pochart P, Machineni H, Welsh M, Kong Y, Zerhusen B, Malcolm R, Varrone Z, Collis A, Minto M, Burgess S, McDaniel L, Stimpson E, Spriggs F, Williams J, Neurath K, Ioime N, Agee M, Voss E, Furtak K, Renzulli R, Aanensen N, Carrolla S, Bickelhaupt E, Lazovatsky Y, DaSilva A, Zhong J, Stanyon CA, Finley RL, White KP, Braverman M, Jarvie T, Gold S, Leach M, Knight J, Shimkets RA, McKenna MP, Chant J, Rothberg JM: A Protein Interaction Map of Drosophila melanogaster. *Science* 2003, 302(5651):1727–1736, [http://dx.doi.org/10.1126/science.1090289].

[7] Babu MM, Luscombe NM, Aravind L, Gerstein M, Teichmann SA: Structure and evolution of transcriptional regulatory networks. *Current opinion in structural biology* 2004, 14(3):283–291, [http://dx.doi.org/10.1016/j.sbi.2004.05.004].

[8] Shen-Orr SS, Milo R, Mangan S, Alon U: Network motifs in the transcriptional regulation network of Escherichia coli. *Nature Genetics* 2002, 31:64–68, [http://dx.doi.org/10.1038/ng881].

[9] Barrios-Rodiles M, Brown KR, Ozdamar B, Bose R, Liu Z, Donovan RS, Shinjo F, Liu Y, Dembowy J, Taylor IW, Luga V, Przulj N, Robinson M, Suzuki H, Hayashizaki Y, Jurisica I, Wrana JL: High-Throughput Mapping of a Dynamic Signaling Network in Mammalian Cells. *Science* 2005, 307(5715):1621–1625, [http://dx.doi.org/10.1126/science.1105776].

[10] Hyduke DR, Palsson B: Towards genome-scale signalling-network reconstructions. *Nature Reviews Genetics* 2010, 11(4):297–307, [http://dx.doi.org/10.1038/nrg2750].

[11] Jeong H, Tombor B, Albert R, Oltvai ZN, Barabasi AL: The large-scale organization of metabolic networks. *Nature* 2000, 407(6804):651–654, [http://dx.doi.org/10.1038/35036627].

[12] Zhang Y, Thiele I, Weekes D, Li Z, Jaroszewski L, Ginalski K, Deacon AM, Wooley J, Lesley SA, Wilson IA, Palsson B, Osterman A, Godzik A: Three-dimensional structural view of the central metabolic network of Thermotoga maritima. *Science (New York, N.Y.)* 2009, 325(5947):1544–1549, [http://dx.doi.org/10.1126/science.1174671].

[13] McDermott JE, Costa M, Janszen D, Singhal M, Tilton SC: Separating the drivers from the driven: Integrative network and pathway approaches aid identification of disease biomarkers from high-throughput data. *Disease markers* 2010, 28(4):253–266, [http://dx.doi.org/10.3233/DMA-2010-0695].

[14] Przytycka TM, Singh M, Slonim DK: Toward the dynamic interactome: it's about time. *Briefings in Bioinformatics* 2010, 11:15–29, [http://dx.doi.org/10.1093/bib/bbp057].

[15] Zhu X, Gerstein M, Snyder M: Getting connected: analysis and principles of biological networks. *Genes & Development* 2007, 21(9):1010–1024, [http://dx.doi.org/10.1101/gad.1528707].

[16] Ideker T, Ozier O, Schwikowski B, Siegel AF: Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics* 2002, 18(suppl 1):S233–S240, [http://dx.doi.org/10.1093/bioinformatics/18.suppl_1.S233].

[17] Barabási ALL, Oltvai ZN: Network biology: understanding the cell's functional organization. *Nature reviews. Genetics* 2004, 5(2):101–113, [http://dx.doi.org/10.1038/nrg1272].

[18] Hawkins RD, Hon GC, Ren B: Next-generation genomics: an integrative approach. *Nature reviews. Genetics* 2010, 11(7):476–486, [http://dx.doi.org/10.1038/nrg2795].

[19] Bitton DA, Okoniewski MJ, Connolly Y, Miller CJ: Exon level integration of proteomics and microarray data. *BMC bioinformatics* 2008, 9:118+, [http://dx.doi.org/10.1186/1471-2105-9-118].

[20] Tsiliki G, Zervakis M, Ioannou M, Sanidas E, Stathopoulos E, Potamias G, Tsiknakis M, Kafetzopoulos D: Multi-platform data integration in microarray analysis. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society* 2011, 15(6):806–812, [http://dx.doi.org/10.1109/TITB.2011.2158232].

[21] Boutros PC: LTR: Linear Cross-Platform Integration of Microarray Data. *Cancer informatics* 2010, 9:197–208, [http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2935818/].

[22] Daigle BJ, Altman RB: M-BISON: microarray-based integration of data sources using networks. *BMC bioinformatics* 2008, 9:214+, [http://dx.doi.org/10.1186/1471-2105-9-214].

[23] Kossinets G: Effects of missing data in social networks. *eprint arXiv:cond-mat/0306335* 2003, [http://arxiv.org/abs/cond-mat/0306335].

[24] Fernandes LP, Annibale A, Kleinjung J, Coolen AC, Fraternali F: Protein networks reveal detection bias and species consistency when analysed by information-theoretic methods. *PloS one* 2010, 5(8):e12083+, [http://dx.doi.org/10.1371/journal.pone.0012083].

[25] Taniguchi Y, Choi PJ, Li GW, Chen H, Babu M, Hearn J, Emili A, Xie XS: Quantifying E. coli Proteome and Transcriptome with Single-Molecule Sensitivity in Single Cells. *Science* 2010, 329(5991):533–538, [http://dx.doi.org/10.1126/science.1188308].

[26] Functional Genomics Data (FGED) Society [http://www.mged.org/].

[27] Taylor CF, Paton NW, Lilley KS, Binz PA, Julian RK, Jones AR, Zhu W, Apweiler R, Aebersold R, Deutsch EW, Dunn MJ, Heck AJR, Leitner A, Macht M, Mann M, Martens L, Neubert TA, Patterson SD, Ping P, Seymour SL, Souda P, Tsugita A, Vandekerckhove J, Vondriska TM, Whitelegge JP, Wilkins MR, Xenarios I, Yates JR, Hermjakob H: The minimum information about a proteomics experiment (MIAPE). *Nature Biotechnology* 2007, 25(8):887–893, [http://dx.doi.org/10.1038/nbt1329].

[28] Orchard S, Salwinski L, Kerrien S, Montecchi-Palazzi L, Oesterheld M, Stumpflen V, Ceol A, Chatr-aryamontri A, Armstrong J, Woollard P, Salama JJ, Moore S, Wojcik J, Bader GD, Vidal M, Cusick ME, Gerstein M, Gavin AC, Superti-Furga G, Greenblatt J, Bader J, Uetz P, Tyers M, Legrain P, Fields S, Mulder N, Gilson M, Niepmann M, Burgoon L, Rivas JD, Prieto C, Perreau VM, Hogue C, Mewes HW, Apweiler R, Xenarios I, Eisenberg D, Cesareni G, Hermjakob H: The minimum information required for reporting a molecular interaction experiment (MIMIx). *Nature Biotechnology* 2007, 25(8):894–898, [http://dx.doi.org/10.1038/nbt1324].

[29] Hoogland C, O'Gorman M, Bogard P, Gibson F, Berth M, Cockell SJ, Ekefjard A, Forsstrom-Olsson O, Kapferer A, Nilsson M, Martinez-Bartolome S, Albar JP, Echevarria-Zomeno S, Martinez-Gomariz M, Joets J, Binz PA, Taylor CF, Dowsey A, Jones AR: Guidelines for reporting the use of gel image informatics in proteomics. *Nature Biotechnology* 2010, 28(7):655–656, [http://dx.doi.org/10.1038/nbt0710-655].

[30] Delmerico JA, Byrnes NA, Bruno AE, Jones MD, Gallo SM, Chaudhary V: Comparing the performance of clusters, Hadoop, and Active Disks on microarray correlation computations. In *2009 International Conference on High Performance Computing (HiPC)*, IEEE 2009:378–387, [http://dx.doi.org/10.1109/HIPC.2009.5433190].

[31] Do Hh, Kirsten T, Rahm E: Comparative Evaluation of Microarray-based Gene Expression Databases. In *Proc. 10. Conf. Database Systems for Business, Technology and Web (BTW* 2003[http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.147.6721].

[32] Jakonien V, Lambrix P: Ontology-based integration for bioinformatics. In *Proceedings of the 31st VLDB Conference* 2005.

[33] Hoehndorf R, Dumontier M, Gennari JH, Wimalaratne S, de Bono B, Cook DL, Gkoutos GV: Integrating systems biology models and biomedical ontologies. *BMC systems biology* 2011, 5:124+, [http://dx.doi.org/10.1186/1752-0509-5-124].

[34] Rosati R: Integrating ontologies and rules: Semantic and computational issues. In *Reasoning Web, Second International Summer School 2006, Lissabon, Portugal, September 25-29, 2006, Tutorial Lectures, volume 4126 of LNCS*, Springer 2006:128–151.

[35] EcoliWiki [http://ecoliwiki.net/colipedia/index.php/Welcome_to_EcoliWiki].

[36] Kawasaki ES: The end of the microarray Tower of Babel: will universal standards lead the way? *Journal of biomolecular techniques : JBT* 2006, 17(3):200–206, [http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2291790/].

[37] Affymetrix [http://www.affymetrix.com/].

[38] Agilent Technologies [www.agilent.com/].

[39] Stekel D: *Microarray Bioinformatics*. Cambridge University Press 2003.

[40] Faith JJ, Driscoll ME, Fusaro VA, Cosgrove EJ, Hayete B, Juhn FS, Schneider SJ, Gardner TS: Many Microbe Microarrays Database: uniformly normalized Affymetrix compendia with structured experimental metadata. *Nucleic Acids Research* 2008, 36(suppl 1):D866–D870, [http://dx.doi.org/10.1093/nar/gkm815].

[41] Ernst J, Beg QK, Kay KA, Balázsi G, Oltvai ZN, Bar-Joseph Z: A semi-supervised method for predicting transcription factor-gene interactions in Escherichia coli. *PLoS computational biology* 2008, 4(3):e1000044+, [http://dx.doi.org/10.1371/journal.pcbi.1000044].

[42] Metzker ML: Sequencing technologies - the next generation. *Nature reviews. Genetics* 2010, 11:31–46, [http://dx.doi.org/10.1038/nrg2626].

[43] Frost [http://www.businesswire.com/news/home/20110622005681/en/Study-Shows-Strong-Future-Demand-Affymetrix-Microarrays].

[44] Edgar R, Domrachev M, Lash AE: Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research* 2002, 30:207–210, [http://dx.doi.org/10.1093/nar/30.1.207].

[45] Wang C, Gevertz J, Chen CC, Auslender L: Finding Important Genes from High-Dimensional Data: An Appraisal of Statistical Tests and Machine-Learning Approaches 2012, [http://arxiv.org/abs/1205.6523].

[46] Dudley JT, Butte AJ: A quick guide for developing effective bioinformatics programming skills. *PLoS computational biology* 2009, 5(12):e1000589+, [http://dx.doi.org/10.1371/journal.pcbi.1000589].

[47] Galperin MY, Fernández-Suárez XM: The 2012 Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection. *Nucleic Acids Research* 2012, 40(D1):D1–D8, [http://dx.doi.org/10.1093/nar/gkr1196].

[48] Covert MW, Knight EM, Reed JL, Herrgard MJ, Palsson BO: Integrating high-throughput and computational data elucidates bacterial networks. *Nature* 2004, 429(6987):92–96, [http://dx.doi.org/10.1038/nature02456].

[49] Thiele I, Jamshidi N, Fleming RMT, Palsson B: Genome-Scale Reconstruction of Escherichia coli's Transcriptional and Translational Machinery: A Knowledge Base, Its Mathematical Formulation, and Its Functional Characterization. *PLoS Comput Biol* 2009, 5(3):e1000312+, [http://dx.doi.org/10.1371/journal.pcbi.1000312].

[50] Smith B, Ashburner M, Rosse C, Bard J, Bug W, Ceusters W, Goldberg LJ, Eilbeck K, Ireland A, Mungall CJ, Leontis N, Rocca-Serra P, Ruttenberg A, Sansone SA, Scheuermann RH, Shah N, Whetzel PL, Lewis S: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology* 2007, 25(11):1251–1255, [http://dx.doi.org/10.1038/nbt1346].

[51] *Formal Ontology in Information Systems: Proceedings of the First International Conference (FOIS'98)* 1998.

[52] Hoehndorf R, Oellrich A, Dumontier M, Kelso J, Rebholz-Schuhmann D, Herre H: Relations as patterns: bridging the gap between OBO and OWL. *BMC bioinformatics* 2010, 11:441+, [http://dx.doi.org/10.1186/1471-2105-11-441].

[53] Harris MA, Clark J, Ireland A, Lomax J, Ashburner M, Foulger R, Eilbeck K, Lewis S, Marshall B, Mungall C, Richter J, Rubin GM, Blake JA, Bult C, Dolan M, Drabkin H, Eppig JT, Hill DP, Ni L, Ringwald M, Balakrishnan R, Cherry JM, Christie KR, Costanzo MC, Dwight SS, Engel S, Fisk DG, Hirschman JE, Hong EL, Nash RS, Sethuraman A, Theesfeld CL, Botstein D, Dolinski K, Feierbach B, Berardini T, Mundodi S, Rhee SY, Apweiler R, Barrell D, Camon E, Dimmer E, Lee V, Chisholm R, Gaudet P, Kibbe W, Kishore R, Schwarz EM, Sternberg P, Gwinn M, Hannick L, Wortman J, Berriman M, Wood V, de la Cruz N, Tonellato P, Jaiswal P, Seigfried T, White R, Gene Ontology Consortium: The Gene Ontology (GO) database and informatics resource. *Nucleic acids research* 2004, 32(Database issue):258D–261, [http://dx.doi.org/10.1093/nar/gkh036].

[54] Courtot M, Juty N, Knüpfer C, Waltemath D, Zhukova A, Dräger A, Dumontier M, Finney A, Golebiewski M, Hastings J, Hoops S, Keating S, Kell DB, Kerrien S, Lawson J, Lister A, Lu J, Machne R, Mendes P, Pocock M, Rodriguez N, Villeger A, Wilkinson DJ, Wimalaratne S, Laibe C, Hucka M, Le Novère N: Controlled vocabularies and semantics in systems biology. *Molecular systems biology* 2011, 7, [http://dx.doi.org/10.1038/msb.2011.77].

[55] Onsongo G, Xie H, Griffin TJ, Carlis J: Generating GO Slim Using Relational Database Management Systems to Support Proteomics Analysis. In *2008 21st IEEE International Symposium on Computer-Based Medical Systems*, IEEE 2008:215–217, [http://dx.doi.org/10.1109/CBMS.2008.77].

[56] Spellman PT, Miller M, Stewart J, Troup C, Sarkans U, Chervitz S, Bernhart D, Sherlock G, Ball C, Lepage M, Swiatek M, Marks WL, Goncalves J, Markel S, Iordan D, Shojatalab M, Pizarro A, White J, Hubley R, Deutsch E, Senger M, Aronow BJ, Robinson A, Bassett D, Stoeckert CJ, Brazma A: Design and implementation of microarray gene expression markup language (MAGE-ML). *Genome biology* 2002, 3(9), [http://www.ncbi.nlm.nih.gov/pmc/articles/PMC126871/].

[57] Codd EF: A relational model of data for large shared data banks. *Commun. ACM* 1970, 13(6):377–387, [http://dx.doi.org/10.1145/362384.362685].

[58] Generic Model Organism Database (GMOD) [http://gmod.org/].

[59] Stein LD, Mungall C, Shu S, Caudy M, Mangone M, Day A, Nickerson E, Stajich JE, Harris TW, Arva A, Lewis S: The generic genome browser: a building block for a model organism system database. *Genome research* 2002, 12(10):1599–1610, [http://dx.doi.org/10.1101/gr.403602].

[60] Podicheti R, Gollapudi R, Dong Q: WebGBrowse–a web server for GBrowse. *Bioinformatics (Oxford, England)* 2009, 25(12):1550–1551, [http://dx.doi.org/10.1093/bioinformatics/btp239].

[61] Ficklin SP, Sanderson LAA, Cheng CHH, Staton ME, Lee T, Cho IHH, Jung S, Bett KE, Main D: Tripal: a construction toolkit for online genome databases. *Database : the journal of biological databases and curation* 2011, 2011, [http://dx.doi.org/10.1093/database/bar044].

[62] Pavlo A, Paulson E, Rasin A, Abadi DJ, DeWitt DJ, Madden S, Stonebraker M: A comparison of approaches to large-scale data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, SIGMOD '09, New York, NY, USA: ACM 2009:165–178, [http://dx.doi.org/10.1145/1559845.1559865].

[63] Mchome M: Comparison Study between MapReduce (MR) and Parallel Data Management Systems (DBMs) in Large Scale Data Analysis. *Master's thesis*, Macalester College 2011.

[64] Gilbert S, Lynch N: Brewer's Conjecture and the Feasibility of Consistent Available Partition-Tolerant Web Services. In *In ACM SIGACT News* 2002:2002.

[65] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE: Bigtable: A distributed storage system for structured data 2006, :205–218, [http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.9822].

[66] HBase [http://hbase.apache.org/].

[67] George L: *Hbase: The Definitive Guide*. O"Reilly Media, Inc. 2011.

[68] Lakshman A, Malik P: Cassandra- A Decentralized Structured Storage System[http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.161.6751].

[69] Hewitt E: *Cassandra: The Definitive Guide*. O"Reilly Media, Inc. 2010.

[70] Groth P, Gil Y: Analyzing the Gap Between Workflows and their Descriptions. In *Proceedings of the IEEE Third International Workshop on Scientific Workflows* 2009.

[71] Goecks J, Nekrutenko A, Taylor J, Team G: Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology* 2010, 11(8):R86+, [http://dx.doi.org/10.1186/gb-2010-11-8-r86].

[72] Oinn T, Addis M, Ferris J, Marvin D, Senger M, Greenwood M, Carver T, Glover K, Pocock MR, Wipat A, Li P: Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 2004, 20(17):3045–3054, [http://dx.doi.org/10.1093/bioinformatics/bth361].

[73] Bylander T, Chandrasekaran B: Generic tasks for knowledge-based reasoning: the “right” level of abstraction for knowledge acquisition. *Int. J. Man-Mach. Stud.* 1987, 26(2):231–243, [http://dx.doi.org/10.1016/S0020-7373(87)80093-7].

[74] Draghici S, Khatri P, Eklund AC, Szallasi Z: Reliability and reproducibility issues in DNA microarray measurements. *Trends in genetics : TIG* 2006, 22(2):101–109, [http://dx.doi.org/10.1016/j.tig.2005.12.005].

[75] Joly S, Stevens MI, van Vuuren BJJ: Haplotype networks can be misleading in the presence of missing data. *Systematic biology* 2007, 56(5):857–862, [http://dx.doi.org/10.1080/10635150701633153].

[76] Seman Kachalo ZA, Liang J: Assessing Effect of Cross-Hybridization on Oligonucleotide Microarrays. In *Methods of microarray data analysis III (Papers from CAMDA'02)* 2002.

[77] Wu C, Carta R, Zhang L: Sequence dependence of cross-hybridization on short oligo microarrays. *Nucleic acids research* 2005, 33(9):e84, [http://dx.doi.org/10.1093/nar/gni082].

[78] Kane MD, Jatkoe TA, Stumpf CR, Lu J, Thomas JD, Madore SJ: Assessment of the sensitivity and specificity of oligonucleotide (50mer) microarrays. *Nucleic Acids Research* 2000, 28(22):4552–4557, [http://dx.doi.org/10.1093/nar/28.22.4552].

[79] Upton G, Langdon W, Harrison A: G-spots cause incorrect expression measurement in Affymetrix microarrays. *BMC Genomics* 2008, 9:613+, [http://dx.doi.org/10.1186/1471-2164-9-613].

[80] Upton GJG, Sanchez-Graillet O, Rowsell J, Arteaga-Salas JM, Graham NS, Stalteri MA, Memon FN, May ST, Harrison AP: On the causes of outliers in Affymetrix GeneChip data. *Briefings in Functional Genomics and Proteomics* 2009, 8(3):199–212, [http://dx.doi.org/10.1093/bfgp/elp027].

[81] Bengtsson H, Jonsson G, Christersson JV: Calibration and assessment of channel-specific biases in microarray data with extended dynamical range. *BMC Bioinformatics* 2004, 5:177+, [http://dx.doi.org/10.1186/1471-2105-5-177].

[82] Shi L, Tong W, Su Z, Han T, Han J, Puri R, Fang H, Frueh F, Goodsaid F, Guo L, Branham W, Chen J, Xu ZA, Harris S, Hong H, Xie Q, Perkins R, Fuscoe J: Microarray scanner calibration curves: characteristics and implications. *BMC Bioinformatics* 2005, 6(Suppl 2):S11+, [http://dx.doi.org/10.1186/1471-2105-6-S2-S11].

[83] Irizarry RA, Hobbs B, Collin F, Beazer-Barclay YD, Antonellis KJ, Scherf U, Speed TP: Exploration, normalization, and summaries of high density

oligonucleotide array probe level data. *Biostat* 2003, 4(2):249–264, [http://dx.doi.org/10.1093/biostatistics/4.2.249].

[84] Wu Z, Irizarry RA, Gentleman R, Martinez-Murillo F, Spencer F: A Model-Based Background Adjustment for Oligonucleotide Expression Arrays. *Journal of the American Statistical Association* 2004, 99(468):909–917, [http://pubs.amstat.org/doi/abs/10.1198/016214504000000683].

[85] Li C, Wong WH: Model-based analysis of oligonucleotide arrays: model validation, design issues and standard error application. *Genome Biology* 2001, 2(8):research0032.1–research0032.11, [http://dx.doi.org/10.1186/gb-2001-2-8-research0032].

[86] Liu, Mei R, Di X, Ryder TB, Hubbell E, Dee S, Webster TA, Harrington CA, Ho, Baid J, Smeekens SP: Analysis of high density expression microarrays with signed-rank call algorithms. *Bioinformatics* 2002, 18(12):1593–1599, [http://dx.doi.org/10.1093/bioinformatics/18.12.1593].

[87] Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JY, Zhang J: Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology* 2004, 5(10), [http://dx.doi.org/10.1186/gb-2004-5-10-r80].

[88] Carter S, Eklund A, Mecham B, Kohane I, Szallasi Z: Redefinition of Affymetrix probe sets by sequence overlap with cDNA microarray probes reduces cross-platform inconsistencies in cancer-associated gene expression measurements. *BMC Bioinformatics* 2005, 6, [http://dx.doi.org/10.1186/1471-2105-6-107].

[89] Dai M, Wang P, Boyd AD, Kostov G, Athey B, Jones EG, Bunney WE, Myers RM, Speed TP, Akil H, Watson SJ, Meng F: Evolving gene/transcript definitions significantly alter the interpretation of GeneChip data. *Nucleic Acids Research* 2005, 33(20):e175, [http://dx.doi.org/10.1093/nar/gni179].

[90] Gautier L, Møller M, Friis-Hansen L, Knudsen S: Alternative mapping of probes to genes for Affymetrix chips. *BMC Bioinformatics* 2004, 5, [http://dx.doi.org/10.1186/1471-2105-5-111].

[91] Harbig J, Sprinkle R, Enkemann SA: A sequence-based identification of the genes detected by probesets on the Affymetrix U133 plus 2.0 array. *Nucleic Acids Research* 2005, 33(3):e31, [http://dx.doi.org/10.1093/nar/gni027].

[92] Kong SW, Hwang KB, Zhang BT, Greenberg SA, Kohane IS, Park PJ: CrossChip: a system supporting comparative analysis of different generations of Affymetrix arrays. *Bioinformatics* 2005, 21(9):2116–2117, [http://dx.doi.org/10.1093/bioinformatics/bti288].

[93] Liu H, Zeeberg BRR, Gang Q, Koru AGG, Ferrucci A, Kahn A, Ryan MCC, Nuhanovic A, Munson PJJ, Reinhold WCC, Kane DWW, Weinstein JNN: AffyProbeMiner: a web resource for computing or retrieving accurately redefined Affymetrix probe sets. *Bioinformatics* 2007, [http://dx.doi.org/10.1093/bioinformatics/btm360].

[94] Mieczkowski J, Tyburczy M, Dabrowski M, Pokarowski P: Probe set filtering increases correlation between Affymetrix GeneChip and qRT-PCR expression measurements. *BMC Bioinformatics* 2010, 11:104+, [http://dx.doi.org/10.1186/1471-2105-11-104].

[95] Sandberg R, Larsson O: Improved precision and accuracy for microarrays using updated probe set definitions. *BMC Bioinformatics* 2007, 8:48+, [http://dx.doi.org/10.1186/1471-2105-8-48].

[96] The NCBI handbook [http://www.ncbi.nlm.nih.gov/books/NBK21101].

[97] Hall JL, Grindle S, Han X, Fermin D, Park S, Chen Y, Bache RJ, Mariash A, Guan Z, Ormaza S, Thompson J, Graziano J, de Sam Lazaro SE, Pan S, Simari RD, Miller LW: Genomic profiling of the human heart before and after mechanical support with a ventricular assist device reveals alterations in vascular signaling networks. *Physiological Genomics* 2004, 17(3):283–291, [http://dx.doi.org/10.1152/physiolgenomics.00004.2004].

[98] Irizarry RA, Warren D, Spencer F, Kim IF, Biswal S, Frank BC, Gabrielson E, Garcia JGN, Geoghegan J, Germino G, Griffin C, Hilmer SC, Hoffman E, Jedlicka AE, Kawasaki E, Martinez-Murillo F, Morsberger L, Lee H, Petersen D, Quackenbush J, Scott A, Wilson M, Yang Y, Ye SQ, Yu W: Multiple-laboratory comparison of microarray platforms. *Nature Methods* 2005, 2(5):345–350, [http://dx.doi.org/10.1038/nmeth756].

[99] Thompson K, Deshmukh H, Solka J, Weller J: A white-box approach to microarray probe response characterization: the BaFL pipeline. *BMC Bioinformatics* 2009, 10:449+, [http://dx.doi.org/10.1186/1471-2105-10-449].

[100] Kumari S, Verma L, Weller J: AffyMAPSDetector: a software tool to characterize Affymetrix GeneChipTM expression arrays with respect to SNPs. *BMC Bioinformatics* 2007, 8:276+, [http://dx.doi.org/10.1186/1471-2105-8-276].

[101] Affymetrix Fusion SDK [http://www.affymetrix.com/partners_programs/programs/developer/fusion/index.affx?terms=no].

[102] Bengtsson H, Bullard J, Gentleman R, Hansen KD, Morgan M: *affxparser: Affymetrix File Parsing SDK* 2009. [R package version 1.16.0].

[103] Data-FATE [http://webpages.uncc.edu/~dcarr10/DataFATE_html/DataFATE_Home.html].

[104] PostgresSQL [http://www.postgresql.org/].

[105] Python [http://www.python.org/].

[106] QT [http://qt.nokia.com/].

[107] PyQt [http://www.riverbankcomputing.co.uk/software/pyqt/intro].

[108] SQLite [http://www.sqlite.org/].

[109] sqlite3 [http://docs.python.org/library/sqlite3.html].

[110] Bhattacharjee A, Richards WG, Staunton J, Li C, Monti S, Vasa P, Ladd C, Beheshti J, Bueno R, Gillette M, Loda M, Weber G, Mark EJ, Lander ES, Wong W, Johnson BE, Golub TR, Sugarbaker DJ, Meyerson M: Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of the National Academy of Sciences of the United States of America* 2001, 98(24):13790–13795, [http://dx.doi.org/10.1073/pnas.191502998].

[111] Irizarry RA, Gautier L, Huber W, Bolstad B: *makecdfenv: CDF Environment Maker* 2006. [R package version 1.22.0].

[112] Huttenhower C, Hibbs M, Myers C, Troyanskaya OG: A scalable method for integration and functional analysis of multiple microarray datasets. *Bioinformatics (Oxford, England)* 2006, 22(23):2890–2897, [http://dx.doi.org/10.1093/bioinformatics/btl492].

[113] Gharaibeh R, Fodor A, Gibas C: Background correction using dinucleotide affinities improves the performance of GCRMA. *BMC Bioinformatics* 2008, 9:452+, [http://dx.doi.org/10.1186/1471-2105-9-452].

[114] Team RDC: *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria 2011, [http://www.R-project.org/]. [ISBN 3-900051-07-0].

[115] Gama-Castro S, Salgado H, Peralta-Gil M, Santos-Zavaleta A, Muñiz Rascado L, Solano-Lira H, Jimenez-Jacinto V, Weiss V, García-Sotelo JS, López-Fuentes A, Porrón-Sotelo L, Alquicira-Hernández S, Medina-Rivera A, Martínez-Flores I, Alquicira-Hernández K, Martínez-Adame R, Bonavides-Martínez C, Miranda-Ríos J, Huerta AM, Mendoza-Vargas A, Collado-Torres L, Taboada B, Vega-Alvarado L, Olvera M, Olvera L, Grande R, Morett E, Collado-Vides J: RegulonDB version 7.0: transcriptional regulation of Escherichia coli K-12 integrated within genetic sensory response units (Gensor Units). *Nucleic Acids Research* 2011, 39(suppl 1):D98–D105, [http://dx.doi.org/10.1093/nar/gkq1110].

[116] Faith JJ, Hayete B, Thaden JT, Mogno I, Wierzbowski J, Cottarel G, Kasif S, Collins JJ, Gardner TS: Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles. *PLoS Biol* 2007, 5:e8+, [http://dx.doi.org/10.1371/journal.pbio.0050008].

[117] Langmead B, Trapnell C, Pop M, Salzberg S: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology* 2009, 10(3):R25+, [http://dx.doi.org/10.1186/gb-2009-10-3-r25].

[118] Li H, Ruan J, Durbin R: Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research* 2008, 18(11):1851–1858, [http://dx.doi.org/10.1101/gr.078212.108].

[119] Liu CM, Wong T, Wu E, Luo R, Yiu SM, Li Y, Wang B, Yu C, Chu X, Zhao K, Li R, Lam TW: SOAP3: Ultra-fast GPU-based parallel alignment tool for short reads. *Bioinformatics* 2012, 28(6):878–879, [http://dx.doi.org/10.1093/bioinformatics/bts061].

[120] Mosaik [http://bioinformatics.bc.edu/marthlab/Mosaik].

[121] Smith TF, Waterman MS: Identification of common molecular subsequences. *Journal of molecular biology* 1981, 147:195–197, [http://view.ncbi.nlm.nih.gov/pubmed/7265238].

[122] Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, Subgroup GPDP: The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 2009, 25(16):2078–2079, [http://dx.doi.org/10.1093/bioinformatics/btp352].

[123] DNA Software [http://dnasoftware.com/].

[124] Gautier L, Cope L, Bolstad BM, Irizarry RA: affy—analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics* 2004, 20(3):307–315.

[125] Clauset A, Newman ME, Moore C: Finding community structure in very large networks. *Physical review. E, Statistical, nonlinear, and soft matter physics* 2004, 70(6 Pt 2), [http://view.ncbi.nlm.nih.gov/pubmed/15697438].

[126] Newman ME: Fast algorithm for detecting community structure in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics* 2004, 69(6 Pt 2), [http://view.ncbi.nlm.nih.gov/pubmed/15244693].

[127] Csardi G, Nepusz T: The igraph software package for complex network research. *InterJournal* 2006, Complex Systems:1695, [http://igraph.sf.net].

[128] Lancichinetti A, Fortunato S: Community detection algorithms: a comparative analysis. *Physical review. E, Statistical, nonlinear, and soft matter physics* 2009, 80(5 Pt 2), [http://view.ncbi.nlm.nih.gov/pubmed/20365053].

[129] Harr B, Schlotterer C: Comparison of algorithms for the analysis of Affymetrix microarray data as evaluated by co-expression of genes in known operons. *Nucleic Acids Research* 2006, 34(2):e8, [http://dx.doi.org/10.1093/nar/gnj010].

[130] Alvarez M, Sumazin P, Rajbhandari P, Califano A: Correlating measurements across samples improves accuracy of large-scale expression profile experiments. *Genome Biology* 2009, 10(12):R143+, [http://dx.doi.org/10.1186/gb-2009-10-12-r143].

[131] Cho BK, Zengler K, Qiu Y, Park YS, Knight EM, Barrett CL, Gao Y, Palsson BO: The transcription unit architecture of the Escherichia coli genome. *Nature biotechnology* 2009, 27(11):1043–1049, [http://dx.doi.org/10.1038/nbt.1582].